Universität St.Gallen

School of Economics and Political Science

# Algorithmic Fairness and Optimal Policy Choice

**Exploring Different Fairness Metrics for Treatment Allocation**

Submitted by

Patrick Leu
21-615-372
Rheingutstrasse 22
CH-8245 Feuerthalen

Supervised by

Prof. PhD David Preinerstorfer

21 May, 2024

## Abstract

Optimally allocating individuals to treatments via elaborate decision rules constitutes a growing phenomenon in various policy areas. When algorithms exclusively aim at maximising welfare, however, they are prone to multiple biases. Hence, different fields of research have been focusing on locating and mitigating the source of those biases. At first, model-based approaches required treatment allocations to adhere to formal mathematical constraints. Only recently, consequentialist frameworks, focusing on the induced outcomes of decisions, became increasingly popular in policy learning. This thesis analyses efficiency-fairness trade-offs that may arise in diverse settings. We implement and compare three methods of treatment allocation, varying in the specific perspective adopted when assigning treatments. Our findings suggest that an outcome-oriented perspective leads to more efficient and more equitable results, underlining the merit of a more consequentialist approach to policy learning.

# Table of Contents

# List of Figures



# List of Tables



# List of Abbreviations

| | |
|---|---|
| AIPW | Augmented Inverse Probability Weighting |
| EWM | Empirical Welfare Maximisation |
| FPT | Fair Policy Targeting |
| IPW | Inverse Probability Weighting |
| MIP | Mixed Integer Program |
| MILP | Mixed Integer Linear Program |
| MIQCP | Mixed Integer Quadratically Constrained Program |
| MIQP | Mixed Integer Quadratic Program |
| SUTVA | Stable Unit Treatment Value Assumption |
| UCB | Upper Confidence Bound |

# 1 Introduction

Increasingly refined statistical methods to estimate heterogeneous treatment effects allowed for the development of decision rules that pinpoint welfare-maximising allocations. Such rules for assigning individuals to specific treatments enjoy growing popularity in various policy areas. In medicine, for instance, algorithms identify and assign patients with complex health needs to specific care management programmes in an effort to reduce costs (Obermeyer et al., 2019). Other examples include training programmes for job seekers (Frölich, 2008), energy rebate programmes to reduce consumption during peak times (Ida et al., 2022), or fostering immigrants' integration by encouraging naturalisation in their host country (Ahrens et al., 2024).

However, concerns that those algorithms may "induce disparities across sensitive attributes, such as age, gender, or race" are increasing (Viviano & Bradic, 2024, p. 730). For instance, consider Fuster et al. (2022) who explore data from the US mortgage market to predict borrowers' default risk using common machine learning models. They show that more sophisticated models achieve higher predictive accuracy and reduce differences in loan approval rates between demographic groups. Yet, the dispersion in required interest rate increases, "both across and within race groups, especially for Black and Hispanic borrowers" (p. 9). Consequently, researchers from various fields devised several definitions of fairness and the corresponding methodology to ensure the results' consistency with them.

Many such definitions relate to a process-oriented perspective on fairness, meaning that they "conceptualise the equity of algorithmic decisions in terms of universal rules (...) rather than considering the consequences of decisions" (Chohlas-Wood et al., 2024, p. 4). In certain policy contexts, however, decision-makers may be unwilling to implement policies that achieve inter-group fairness yet simultaneously reduce overall welfare. Instead, they prefer a trade-off between efficiency – that is, higher social welfare through e.g., better health status or increased labour force participation rates – and fairness, such as less dispersed acceptance rates across demographic groups (Viviano & Bradic, 2024, p. 730). As a result, policy learning methods which focus on the outcomes of actions may be more appropriate than imposing rigid fairness principles in circumstances that allow for such trade-offs.

We examine three different approaches to the design of treatment allocation rules: (1) Viviano and Bradic's (2024) fair policy targeting (FPT) algorithm that first locates Pareto-optimal policies and subsequently chooses the fairest allocation; (2) a regularised approach that penalises disparities between demographic groups inspired by Chohlas-Wood et al. (2024) and Kock and Preinerstorfer (2024); and (3) empirical welfare maximisation (EWM) such as in Kitagawa and Tetenov (2018). We apply those methods to data derived from the National Study of Learning Mindsets – an educational programme designed to promote student academic achievement – and compare their impact on welfare and fairness outcomes.

The rest of the thesis is organised as follows. Section 2 contains a brief review of the relevant literature, notably contrasting axiomatic and consequentialist perspectives. Section 3 introduces the methodologies of treatment allocation specified above. Section 4 presents our empirical application. We describe the data, elaborate on the estimation methodology, and discuss the results and challenges experienced during the process. Section 5 concludes.

# 2 Relevant Literature

This paper builds on the literature on classification, statistical treatment rules, the estimation of treatment effects, and algorithmic fairness, briefly discussed below.

The growing literature on statistical treatment rules attempts to assign individuals to treatments in a way that minimises the maximum regret (Ahrens et al., 2024; Athey & Wager, 2021; Hirano & Porter, 2009; Kitagawa & Tetenov, 2018; Manski, 2004; Tetenov, 2012; Zhou et al., 2023). In this context, regret is defined in terms of the difference between the maximal welfare that policymakers could hypothetically achieve given full knowledge about both – treated and untreated – potential outcomes and the expected welfare derived from a certain decision rule given the observed outcomes (Tetenov, 2012, p. 157). Minimising the maximum regret then yields “the least upper bound on the loss in expected welfare that results from not knowing the states of nature” (Manski, 2004, p. 1228). Accordingly, the estimation of (heterogeneous) treatment effects is central to the endeavour of designing optimal decision rules. For instance, see Athey and Imbens (2016) and Athey and Wager (2019) who consider the estimation of causal effects in observational studies.

Manski (2004) points to the similarity between designing a policy function that allocates individuals to a certain treatment and conventional classification tasks in machine learning, in the sense that the policy function or assignment rule seeks to assign individuals to treatments based on their covariates (p. 1222). However, it differs in so far as we only observe an individual’s treated or untreated outcome instead of the correct classification, and in that this outcome may be real-valued (Kitagawa & Tetenov, 2018, p. 595). Additionally, the policy function often abides by certain additional constraints, such as capacity or budget restrictions, as treatments are often costly (Athey & Wager, 2021, p. 133).

However, with the increasing usage of statistical rules in decision-making, concerns have been raised about human biases being encoded and perpetuated via those algorithms (Chouldechova & Roth, 2018; Corbett-Davies et al., 2023; Cowgill & Tucker, 2020; Verma & Rubin, 2018). To tackle potential biases, researchers in the field have focused on two distinct paths: how to detect and correct for bias within the data, and how to develop fair machine learning models.

The biases encoded within the data that could influence subsequent predictions or decisions are often a result of unrepresentative training samples, stemming from past human-made decisions, including taste-based (individual biases and preferences) or statistical (group averages and imperfect information) discrimination (Cowgill & Tucker, 2020, p. 3). An algorithm fitted to such distorted data is likely to codify and perpetuate an already present bias, and accordingly understanding the root of the bias and how to correct it constitutes a core issue in algorithmic fairness (Chouldechova & Roth, 2018, p. 6).[1]

Model-based approaches to reduce bias or unfairness in algorithms stem to a large part from the computer science or machine learning community, and work with formal fairness criteria, which show properties that are considered desirable and that outcomes or predictions should adhere to (Chohlas-Wood et al., 2024; Corbett-Davies et al., 2023; Gupta et al., 2020; Verma & Rubin, 2018; Viviano & Bradic, 2024). Verma and Rubin (2018) differentiate between statistical, individual, and causal fairness metrics. The first class comprises notions of fairness that try to limit disparities between demographic groups, i.e., they apply to the respective group's average individual (Chouldechova & Roth, 2018, p. 3; Corbett-Davies et al., 2023, p. 4). Examples include parity in predictions or equalised odds (see e.g., Hardt et al., 2016, p. 3). Individual notions of fairness involve an approach often referred to as «fairness through unawareness», which consists of blinding algorithms to sensitive attributes such that they do not affect outcomes (Corbett-Davies et al., 2023, p. 9; Cowgill & Tucker, 2020, p. 20). However, as Corbett-Davies et al. (2023) maintain, the sensitive attribute may still indirectly influence the result through its effect on the individuals' other characteristics (p. 10), which led researchers to develop causal paths and counterfactual versions of blinding to control as well for those indirect effects (see e.g., Kusner et al., 2017; Nabi et al., 2019).

[1]However, as Cowgill and Tucker (2020) argue, unrepresentative data is not an insurmountable problem. Needless to say, non-randomly missing data and selection effects potentially lead to biased predictions, however, given enough noise in human decision-making that allows the algorithm to explore different options and the programmer's awareness of working with unrepresentative data, algorithms could reduce the bias encoded in historical, human-made decisions (pp. 4–5). Moreover, the biases in algorithms are more easily detectable and malleable since they are documented in the code, the inputs, and the outcomes, as opposed to the bias in human-made decisions (p. 14).

However, these axiomatic approaches to fairness – that impose formal statistical criteria and thereby constrain the decision space – do not yield better decisions. In contrast, Corbett-Davies et al. (2023), Kleinberg et al. (2018), and Nilforoshan et al. (2022) show that they lead to less efficient *and* less equitable outcomes. For instance, Nilforoshan et al. (2022) demonstrate that in the context of student-body diversity and college admissions various definitions of fairness lead to worse outcomes on both dimensions, meaning that those constraints "may even harm the very groups they were ostensibly designed to protect" (p. 7).[2]

As Chohlas-Wood et al. (2024, p. 2) put it, the "traditional axiomatic approaches to fairness typically do not consider the downstream consequences of constraints, and thus fail to engage with the difficult trade-offs at the heart of many policy problems" (p. 2). Consequently, recent developments – notably influenced by an economic perspective on fairness – concentrated on fairness as part of the decision-maker's preferences rather than imposing ancillary constraints on the decision space (Rambachan, Kleinberg, Ludwig, & Mullainathan, 2020, p. 93). Such consequentialist approaches to algorithmic decision-making take the present trade-offs directly into consideration, with a focus on the Pareto optimality of the policy function and corresponding outcomes (Viviano & Bradic, 2024, p. 730). The specific configuration of the decision problem then may differ in whether the policymaker first singles out the set of Pareto efficient policies – that is, the set of allocations where it is not possible to make one group better off without making another one worse off – and subsequently selects the fairest one according to the adopted preferences (see e.g., Viviano & Bradic, 2024), or if the fairness preferences are directly implemented into the policymaker's utility function (see e.g., Chohlas-Wood et al., 2024; Kock & Preinerstorfer, 2024).

[2] Nilforoshan et al.'s (2022) findings apply to "natural families of utility functions" including those that exhibit a preference for more diversity and higher academic achievement (p. 2). However, especially in the US, decision-makers may be legally prevented from using sensitive attributes in certain policy areas, e.g., staff recruitment (Corbett-Davies et al., 2023, p. 31), and, by implication, potentially relevant information remains unused. And yet a decision-maker adopting a "meta-consequentialist position" could reason that procedural considerations such as the omission of sensitive attributes "engender trust [in the fairness of the decision] and in turn bring better downstream outcomes" (Corbett-Davies et al., 2023, p. 33). In this paper, however, we focus on the outcome-oriented consequentialist perspective.

# 3 Fair Policy Learning

We begin this section by introducing the general problem statement and the conventional notation in the policy learning literature. Specifically, we consider the setting of binary treatments, although this restriction can be easily lifted to include multi-action policies as well (see e.g., Ahrens et al., 2024; Chohlas-Wood et al., 2024; Zhou et al., 2023). We will continue to describe two different methodologies that assign policies or treatments to individuals under certain fairness considerations from a consequentialist perspective, namely the FPT method of Viviano and Bradic (2024) in Section 3.2, and a regularised optimisation approach largely based on Chohlas-Wood et al. (2024) and Kock and Preinerstorfer (2024) in Section 3.3.

## 3.1 General Problem Statement

Consider a random sample of individuals, where each individual $i \in \{1, \ldots, n\}$ exhibits $p$ observable characteristics $X_i$ drawn from the covariate space $\mathcal{X} \subseteq \mathbb{R}^p$. In addition, individuals possess a binary sensitive attribute $S_i \in \{0, 1\}$, where $S = 1$ refers to the disadvantaged group. Each individual $i$ is subject to a binary treatment $D_i \in \{0, 1\}$. Following the potential outcome notation (Imbens & Rubin, 2015, p. 6), $Y_i(d)$ for $d \in \{0, 1\}$ denotes individual $i$'s potential outcomes under treatment $d$, where the outcome $Y_i$ belongs to the outcome space $\mathcal{Y} \in \mathbb{R}$. We denote as $Y_i(D_i)$ the effectively observed outcome.

The setup imposes the following assumptions (Viviano & Bradic, 2024, pp. 732, 735; Zhou et al., 2023, p. 9):

**Assumption 3.1:** Treatment Unconfoundedness. $Y_i(d) \perp D_i \mid X_i, S_i, \ \forall d \in \{0, 1\}$
**Assumption 3.2:** Bounded outcomes. $-\infty < Y_i(d) < \infty, \ \forall d \in \{0, 1\}$

The first assumption states that conditioned on the individual's characteristics and sensitive attribute, its potential outcomes do not depend on the assigned treatment. Assuming for unconfoundedness is standard in the literature, however, as it lacks testability in the observational setting it is not an uncontested assumption (Kitagawa & Tetenov, 2018, p. 597). While the second assumption is not essential, it reflects the realistic setting of many empirical applications and facilitates proofs and notation (Zhou et al., 2023, p. 9).

Given i.i.d. tuples of $(X_i, S_i, D_i, Y_i)$, the ambition is to design a (deterministic) policy function $\pi$ that assigns individuals to a treatment based on its characteristics. Formally, we have $\pi : \mathcal{X} \times S \mapsto D = \{0, 1\}$, where $\pi \in \Pi$ and $\Pi$ is the decision space (or policy function class) that includes potential legal and/or economic constraints (Viviano & Bradic, 2024, p. 732; Athey & Wager, 2021, p. 138).

## 3.2 Fair Policy Targeting

Building on the general setup above, in this section we present the FPT methodology put forward by Viviano and Bradic (2024). Their approach consists of a two-step procedure: First, they maximise the different groups' expected welfare while imposing Pareto efficiency on the selected policies. That is, policy $\pi_1$ is (weakly) preferred to policy $\pi_2$ if the welfare each group derives from $\pi_1$ is (at least as large) larger than the welfare derived from $\pi_2$. Second, from the set of Pareto optimal policies, the FPT algorithm selects – based on distinct fairness criteria presented further below – the policy that ensures the fairest outcome between groups (p. 733).[3] All the proofs for the respective propositions can be found in Viviano and Bradic (2024, Appendix A)

### 3.2.1 Decision Problem

The welfare that individuals from the group $S = s$ derive from a policy function $\pi$ corresponds to the conditional average treatment effect and can be denoted by:

$$W_s(\pi) = \mathbb{E}\Big[\Big(Y(1) - Y(0)\Big)\pi(X, S) \mid S = s\Big] \tag{1}$$

This notation is consistent with an «intention to treat perspective», meaning that individuals assigned to treatment are considered as treated, independent of whether they effectively complied with their assignment (Kitagawa & Tetenov, 2018, p. 608).

[3] These preferences correspond to the assumption of lexicographic and rational preferences, i.e., preferences that fulfil the axioms of completeness and transitivity (Bonanno, 2017, p. 83).

Next, to maximise overall welfare, we take a weighted combination of the two groups' welfare. That is, we multiply each group's welfare with a weighting factor $\alpha$ that denotes the relative importance weight of the group with $S = 0$. In the case of empirical welfare maximisation (EWM), the weighting factor is determined based on the proportion of the sensitive attribute in the sample population. However, this may result in disproportionately adverse results for the minority group (Viviano & Bradic, 2024, p. 732). Consequently, the method we present does not rely on some predefined weights to maximise welfare. Instead, it explores a range of weights $\alpha \in (0, 1)$, and identifies an optimal policy $\pi_\alpha$ for each possible weighting factor. All of them combined define the set of Pareto optimal policies, i.e., the Pareto frontier $\Pi_0 \subseteq \Pi$, formally denoted as (Viviano & Bradic, 2024, Lemma 2.1):

$$\Pi_o = \left\{ \pi_\alpha : \pi_\alpha \in \arg\sup_{\pi \in \Pi} \ \alpha W_0(\pi) + (1-\alpha) W_1(\pi), \text{ where } \alpha \in (0,1) \right\} \tag{2}$$

In the second step, the respective fairness of each policy is assessed by a certain unfairness measure $\mathcal{V}(\pi)$, which, on a generic level, maps each policy to a real number as its associated unfairness level. Section 3.2.3 gives two examples of such unfairness metrics. Combining the two steps, the decision maker faces the following problem (Viviano & Bradic, 2024, Proposition 2.2):

$$\begin{aligned}
\pi^* &\in \arg\inf_{\pi \in \Pi} \mathcal{V}(\pi) \\
\text{s.t. } \bar{W}_\alpha &\leq \alpha W_0(\pi) + (1-\alpha) W_1(\pi), \ \forall \alpha \in (0,1) \\
\text{where } \bar{W}_\alpha &= \sup_{\pi \in \Pi} \ \alpha W_0(\pi) + (1-\alpha) W_1(\pi)
\end{aligned} \tag{3}$$

The optimal policy $\pi^*$ is the one with minimal unfairness subject to it being Pareto optimal. As Viviano and Bradic (2024) show, the advantage of the proposed method is that the welfare maximisation does not rely on "some *pre-specified* and hard-to-justify weights" but instead that "each group's importance (...) is implicitly chosen within the optimisation problem" so as to minimise unfairness (p. 733). This yields better outcomes in terms of fairness and efficiency than alternative approaches which use predetermined importance weights or impose additional fairness constraints on the decision space (see e.g., Fang et al., 2023; Kitagawa & Tetenov, 2018; Nabi et al., 2019; Rambachan, Kleinberg, Mullainathan, & Ludwig, 2020).

### 3.2.2 Estimation of Optimal Fair Policies

We start the construction of an estimator for the optimal policy $\pi^*$ with the estimation of the group-specific welfare. Denote the individuals' expected conditional outcome, given that they received treatment $D_i = d$, and belong to group $S_i = s$, as in Equation 4. This is also referred to as the outcome model. However, due to potential misspecification, directly estimating the treatment effect or welfare from this moment function will likely result in a biased estimation (Chernozhukov et al., 2022, p. 1502). Accordingly, further elements and assumptions need to be introduced.

$$m_{d,s}(x) = \mathbb{E}\left[Y_i(d) \mid X_i = x, S_i = s\right] \tag{4}$$

On the left-hand side of Equation 5, we define the propensity score as the probability of being assigned to treatment $D = d$, conditioned on covariates and the sensitive attribute. On the right-hand side, we denote the probability of belonging to group $S = s$ (Viviano & Bradic, 2024, p. 732):

$$e_d(x, s) = P(D = d|X = x, S = s), \quad p_s = P(S = s) \tag{5}$$

Imposing the following assumption on the propensity score is standard in the policy learning literature (Ahrens et al., 2024, p. 6; Viviano & Bradic, 2024, p. 735; Zhou et al., 2023, p. 9):

**Assumption 3.3:** Overlap. $e_d(x, s), p_s \in (\delta, 1-\delta)$, $\forall \delta \in (0, 1)$ and $\forall (x, s) \in \mathcal{X} \times S$

Rosenbaum and Rubin (1983) demonstrate that unbiased estimates of the average treatment effect – and hence also of the welfare as defined in Equation 1 – can be constructed in the setting of randomised experiments (pp. 46-47), which inherently provide known propensity scores and satisfy Assumptions 3.1 and 3.3 by design (Kitagawa & Tetenov, 2018, p. 597). An example of such an unbiased estimator is the inverse probability weighting (IPW) estimator as used by Kitagawa and Tetenov (2018, p. 593), to estimate the resulting welfare in the setting of a randomised control trial (RCT).

However, as mentioned above, notably unconfoundedness is often disputed in the observational setting. An approach to mitigate this issue is the introduction of doubly robust estimators, which build on the semi-parametric estimation literature (Robins & Rotnitzky, 1995; Robins et al., 1994). Doubly robust estimators combine the (inverse) propensity score and the expected conditional outcome to estimate the resulting welfare, and allow for an unbiased estimation thereof if either of the above nuisance parameters is consistently estimated (Chernozhukov et al., 2022, p. 1521; Zhou et al., 2023, p. 10).[4] Viviano and Bradic (2024) define the doubly robust score of individual $i$ with treatment $D_i = d$ and group membership $S_i = s$ as in Equation 6 (p. 734), which largely corresponds to the augmented inverse probability weighting (AIPW) estimator used by Athey and Wager (2021):

$$\Gamma_{d,s,i} = \frac{\mathbf{1}\{S_i = s\}}{p_s} \left[ \frac{\mathbf{1}\{D_i = d\}}{e_d(X_i, S_i)} (Y_i - m_{d,s}(X_i)) + m_{d,s}(X_i) \right]. \tag{6}$$

The empirical counterpart uses the estimates $\hat{e}_d(X_i, S_i)$ and $\hat{m}_{d,s}(X_i)$ that are constructed through cross-fitting. This involves dividing the data into $K$ different folds. Then, for each fold, $K - 1$ folds are used to estimate the nuisance functions, which are then used to predict the target for the remaining fold (Viviano & Bradic, 2024, Appendix B.2; see also Zhou et al., 2023, p. 11). Used in this manner, cross-fitting reduces the own-observation bias (Ahrens et al., 2024, p. 8; Chernozhukov et al., 2022, p. 1507). This in turn allows for simplified assumptions on the nuisance parameters that guarantee the asymptotic consistency at $\frac{1}{\sqrt{n}}$–rate of the doubly robust estimator (Chernozhukov et al., 2018, p. 10; Chernozhukov et al., 2022, p. 1526).[5]

[4]Chernozhukov et al. (2022) further use orthogonal moment functions in the construction of doubly robust estimators, which are insensitive to local perturbation around the true value of the parameter of interest and therefore additionally reduce the bias of the estimation (p.1502).

[5]Simplification of the assumptions entails, according to Chernozhukov et al. (2022, p. 1526), the omission of the Donsker conditions. The simplified assumptions then require, as stated in Viviano and Bradic (2024, p. 735), that the nuisance parameters converge in probability to their true values at the parametric rate.

Finally, the estimated welfare generated by policy $\pi$ for individuals affiliated with group $S = s$ is defined as the difference between their treated and untreated doubly robust scores:

$$\hat{W}_s(\pi) = \frac{1}{n} \sum_{i=1}^{n} \left( \hat{\Gamma}_{1,s,i} - \hat{\Gamma}_{0,s,i} \right) \pi(X_i, s) \tag{7}$$

The estimation of the Pareto frontier follows a two-step approximation procedure (Viviano & Bradic, 2024, p. 734). First, it is necessary to discretise the group weighting factors $\alpha$, by selecting $N$ equally spaced values within the unit interval.[6] Second, linear constraints are used to characterise the set, by first identifying the largest estimated welfare that can be achieved using each weighting factor $\alpha_j$ as in Equation 8, and then forcing the effectively estimated welfare to be at least as large as $\bar{W}_{j,n}$, up to a small slackness parameter $\frac{\lambda}{\sqrt{n}}$. The approximated Pareto frontier then reads as in Equation 9:

$$\bar{W}_{j,n} = \sup_{\pi \in \Pi} \left\{ \alpha_j \hat{W}_0(\pi) + (1 - \alpha_j) \hat{W}_1(\pi) \right\}, \ \forall j \in \{1, \ldots, N\}, \tag{8}$$

$$\begin{aligned} \hat{\Pi}_o(\lambda) = \Big\{ & \pi \in \Pi : \text{ there is } j \in \{1, \ldots, N\} \\ & \text{s.t. } \alpha_j \hat{W}_0(\pi) + (1 - \alpha_j) \hat{W}_1(\pi) \geq \bar{W}_{j,n} - \frac{\lambda}{\sqrt{n}} \Big\} \end{aligned} \tag{9}$$

Viviano and Bradic (2024) show that the approximate Pareto frontier indeed contains all optimal solutions with a high probability and that it converges to its population counterpart defined in Equation 2 at the parametric $\frac{1}{\sqrt{n}}$–rate (Theorem 4.1 and Theorem 4.2).

To efficiently compute the optimal policy $\hat{\pi}_\lambda$ we represent the optimisation problem as a mixed integer program (MIP), requiring two additional specifications. First, we introduce $n$ binary decision variables $z_{s,i} = \pi(X_i, s)$ that denote individual $i$'s treatment assignment given the decision rule $\pi$ and that it's sensitive attribute is $S = s$ (Florios & Skouras, 2008, p. 87; Viviano & Bradic, 2024, p. 734). Second, we define a vector $\mathbf{u} = \{u_1, \ldots, u_N\}$ counting a total of $N$ binary variables, which helps

[6] The choice of $N$ is arbitrary, however, as Viviano and Bradic (2024) note, the weights need to be equally spaced (p. 734). In the later empirical application, we follow them in using $N = \sqrt{n}$.

to ensure that the estimated solution belongs to the Pareto frontier as in Equation 9. The individual components $u_j$ are used to mark the position on the discretised weight grid where the conditions of the Pareto frontier are met, with $u_j = 1$ indicating that $\alpha_j$ satisfies the criterion. Finally, combining all elements then yields the decision maker's full optimisation problem (Viviano & Bradic, 2024, p. 735):

$$\hat{\pi}_\lambda \arg \min_{\pi, \mathbf{z_0}, \mathbf{z_1}, \mathbf{u}} \hat{\mathcal{V}}(\pi) \quad \text{subject to} \tag{10}$$

$$z_{s,i} = \pi(X_i, s), \quad 1 \leq i \leq n, \tag{A}$$

$$u_j \alpha_j \langle \hat{\mathbf{\Gamma}}_{1,0} - \hat{\mathbf{\Gamma}}_{0,0}, \mathbf{z}_0 \rangle + u_j(1-\alpha_j)\langle \hat{\mathbf{\Gamma}}_{1,1} - \hat{\mathbf{\Gamma}}_{0,1}, \mathbf{z}_1 \rangle \geq u_j n \bar{W}_{j,n} - \sqrt{n}\lambda \tag{B}$$

$$\langle \mathbf{1}, \mathbf{u} \rangle \geq 1, \tag{C}$$

$$u_j \in \{0, 1\}, \quad 1 \leq j \leq N \tag{D}$$

$$\pi \in \Pi \tag{E}$$

The linear constraint in (A) represents the additional $n$ binary decision variables introduced for the MIP representation. Constraints (B) and (C) enforce Pareto optimality. The vectors $\mathbf{\Gamma}_{d,s}$ and $\mathbf{z}_s$ denote the aggregated individuals' doubly robust scores and treatment assignments, respectively, and the function $\langle \cdot, \cdot \rangle$ denotes the inner product. For the policy to be Pareto optimal, the condition needs to be fulfilled for at least one $\alpha_j$, meaning that there must exist at least one $u_j \neq 0$, and, consequently, the sum of the individual $u_j$ must add up to at least one. In this representation, the optimisation problem constitutes a mixed integer quadratically constrained problem (MIQCP) and in the case of a non-linear objective function $\hat{\mathcal{V}}$ a mixed integer quadratic program (MIQP). Constraint (D) enforces the individual $u_j$ to be binary, and constraint (E) requires the policy function to belong to the respective function class.

### 3.2.3 Fairness Metrics

This Section presents two fairness metrics that Viviano and Bradic (2024) introduce, one relating to distributional and the other to counterfactual notions of fairness. The first metric, prediction disparity, corresponds to the former idea, and "captures disparity in the treatment [assignment] probability between groups" (p.

736). The policy $\pi$ is deterministic, meaning it is binary, and consequently taking the expectation yields the probability of the treatment being assigned. Conditioning on the sensitive attribute gives the conditional assignment probabilities, which are subtracted to form the unfairness measure as in Equation 11, and its corresponding estimator in Equation 12. In this setting, minimising unfairness implies maximising the treatment assignment probability of the disadvantaged group.

$$\mathcal{V}(\pi) = \mathbb{E}\Big[\pi(X,S)|S=0\Big] - \mathbb{E}\Big[\pi(X,S)|S=1\Big] \tag{11}$$

$$\hat{\mathcal{V}}(\pi) = \frac{1}{n}\sum_{i=1}^{n}\frac{\pi(X_i)(1-S_i)}{(1-\hat{p}_1)} - \frac{1}{n}\sum_{i=1}^{n}\frac{\pi(X_i)S_i}{\hat{p}_1} \tag{12}$$

When taking the absolute value of the prediction disparity metric, none of the two groups is favoured and the objective $|\hat{\mathcal{V}}(\pi)|$ instead minimises the difference in treatment assignment probabilities between the respective groups.

The counterfactual notion of fairness is captured by the concept of envy freeness (p. 738; Appendix B.4.2). Envy freeness represents a state where each agent feels that her own allocation is at least as good as any other agent's allocation (Varian, 1976, Abstract), and requires the following additional assumptions:

**Assumption 3.4:** $Y_i(d,s) \perp (D_i,S_i) \mid X_i(s),\ \forall d,s \in \{0,1\}$
**Assumption 3.5:** $X_i(s) \perp S_i,\ \forall s \in \{0,1\}$

While Assumption 3.4 requires an individual's potential outcomes to be independent of its treatment assignment (unconfoundedness) *and* its group affiliation, conditioned on the potential covariates, Assumption 3.5 requires independence of the potential covariates and the protected attribute (Viviano & Bradic, 2024, p. 738). Observed outcomes and covariates, however, may depend on the sensitive attribute. To construct a measure of envy, consider first the welfare effect of a policy designed for individuals who are affiliated with group $S = s_1$ on members of group $S = s_2$, conditioned on covariates:[7]

[7] The subsequent notation departs from Viviano and Bradic (2024) in so far as it leaves out the baseline effect, which is not affected by the counterfactual policy (p. 738).

$$V_{\pi(X_i,s_1)}(X_i,s_2) = \mathbb{E}\Big[\Big(Y_i(1,s_2) - Y_i(0,s_2)\Big)\pi(X_i,s_1)\Big|\, X_i(s_2) = x\Big] \tag{13}$$

The envy of an individual pertaining to group $S = s_2$ is then measured as the difference between her expected conditional welfare but with the policy and covariate functions of the opposite group (counterfactual welfare; first expectation in Equation 14), and her «true» expected conditional welfare with corresponding policy and covariate functions (second expectation in Equation 14). An appropriate estimator is defined below in Equation 15.

$$\mathcal{A}(s_1,s_2;\pi) = \mathbb{E}_{X(s_1)}\Big[V_{\pi(X(s_1),s_1)}(X(s_1),s_2)\Big] - \mathbb{E}_{X(s_2)}\Big[V_{\pi(X(s_2),s_2)}(X(s_2),s_2)\Big] \tag{14}$$

$$\begin{aligned}\hat{\mathcal{A}}(s_1,s_2;\pi) = &\frac{1}{n}\sum_{i=1}\frac{\mathbf{1}\{S_i = s_1\}}{\hat{p}_{s_1}}\Big[\Big(\hat{m}_{1,s_2}(X_i) - \hat{m}_{0,s_2}(X_i)\Big)\pi(X_i,s_1)\Big] \\ &- \frac{1}{n}\sum_{i=1}^{n}\Big[(\hat{\Gamma}_{1,s_1,i} - \hat{\Gamma}_{0,s_1,i})\pi(X_i,s_1)\Big]\end{aligned} \tag{15}$$

Ultimately, to get a measure of unfairness that does not favour either group, the envies of the respective groups are added up, yielding $\hat{\mathcal{V}}(\pi) = \hat{\mathcal{A}}(0,1;\pi) + \hat{\mathcal{A}}(1,0;\pi)$ as the objective function in the optimisation problem (Viviano & Bradic, 2024, Appendix B.4.2).

## 3.3 Regularised Optimisation

In this section, we introduce the regularised optimisation approach for learning fair policies, largely based on the methods proposed by Chohlas-Wood et al. (2024) and Kock and Preinerstorfer (2024). One of the main contributions of the two works is the strong focus on tackling the problem from a consequentialist perspective and, accordingly, on the decision-maker's diverse preferences. In contrast to other common approaches, they combine efficiency and fairness considerations within the same utility function instead of imposing one of the two policy objectives via constraints. In Chohlas-Wood et al. (2024), this allows the authors to draw a Pareto frontier where each point corresponds to a different set of preferences, which in turn reflect the context-specific trade-off between efficiency and fairness. Moreover, to illustrate the trade-off empirically, they designed a representative poll in which the evaluation of the responses showed that a majority of the participants indeed preferred "trading at least some efficiency" to get more equitable outcomes (p. 10).[8] To benefit from adaptive online learning in the context of exploration-exploitation, Chohlas-Wood et al. (2024) use contextual bandit algorithms such as $\epsilon$-greedy, Thompson sampling or upper confidence bound (UCB) algorithms to estimate the resulting welfare (p. 21).[9] In our approach, we abstract from adaptive learning but combine the methods of EWM with the proposed focus on the policymaker's preferences.

### 3.3.1 Decision Problem

As a starting point, consider the general policy learning setup as laid out in Section 3.1. From there on, the utility policy-makers derive from a policy $\pi$ depends on two factors: first, on the reward an individual $i$ obtains from the regime, and second, on the policymaker's fairness preferences. The latter are encoded in the utility function as penalties on inequitable allocations, which are defined correspondingly by said preferences. In the context of a binary sensitive attribute, this yields the following utility function (Chohlas-Wood et al., 2024, p. 11):

$$\begin{aligned} U(\pi) = \mathbb{E}_{X,Y}\Big[r\Big(X, \pi(X), Y(\pi(X))\Big)\Big] \\ - \sum_{\ell=1}^{L} \lambda_\ell \Big| \mathbb{E}_{X,Y}\left[f_\ell\Big(X, \pi(X), Y(\pi(X))\Big) \mid S = s_1\right] \\ - \mathbb{E}_{X,Y}\left[f_\ell\Big(X, \pi(X), Y(\pi(X))\Big) \mid S = s_2\right]\Big| \end{aligned} \tag{16}$$

[8]See also Koenecke et al. (2023) who, in the context of a food programme, show that community preferences allow for less efficiency in exchange for a more equal distribution among ethnicities.

[9]Lattimore and Szepesvári (2020) define a bandit problem as "a sequential game between a learner and an environment" (p. 10). The learner (or decision-maker) has to choose between multiple actions, each with an unknown reward structure, with the goal of maximising the cumulative reward generated over all rounds played. The challenge for the decision-maker is to find a policy rule that balances between exploiting high-reward actions and exploring other actions to gain further knowledge about the reward distributions. Contextual bandits then – as the name suggests – not only make use of the learned reward structure but also include additional information available within the decision's context, improving the quality of the chosen policy (p. 224). For instance, contextual bandits may take an individual's demographic information or preferences (context) into account when deciding on which advertisement (action) to show them to maximise the probability that the individual clicks on it (reward).

While the function $r(\cdot)$ denotes the reward, $f_\ell(\cdot)$ where $\ell \in \{1, \dots, L\}$ encapsulate the policy-maker's fairness preferences. Examples for $f_\ell$ include cost functions so as to ensure equal spending on individuals from each group, or treatment assignment probabilities to achieve equal assignment rates. The coefficient $\lambda_\ell$ regulates the strength of the penalty, and $|\cdot|$ denotes the absolute value.

For the method we propose, the welfare as defined in Equation 1 represents the reward for each group. The two welfares are then added up, and, as in the EWM approach, weighted by the respective groups' share in the population. Moreover, the policy-makers show a preference for equal treatment assignment probabilities, as in Equation 11. Thus, the adjusted utility function can be written as in Equation 17, and maximising this utility yields the optimal policy $\pi^* \in \arg\max_\pi U(\pi)$.

$$U(\pi) = \Big[p_0 W_0(\pi) + p_1 W_1(\pi)\Big] - \lambda \Big|\mathbb{E}\Big[\pi(X,S) \mid S=0\Big] - \mathbb{E}\Big[\pi(X,S) \mid S=1\Big]\Big| \tag{17}$$

### 3.3.2 Estimation of Optimal Fair Policies

For the construction of an estimator of the utility function, we resort to the doubly robust scores as estimators for the welfare (Equation 7) and use the estimator of the prediction disparity unfairness measure for the treatment assignment probabilities (Equation 12). This gives the following estimator:

$$\hat{U}(\pi) = \hat{p}_0 \hat{W}_0(\pi) + \hat{p}_1 \hat{W}_1(\pi) - \lambda \Big| \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(X_i)(1-S_i)}{\hat{p}_0} - \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(X_i) S_i}{\hat{p}_1} \Big|$$

To represent the optimisation problem as a MIP, we again introduce $n$ binary decision variables $z_{s,i} = \pi(X_i, s)$ that denote an individual's treatment assignment. Moreover, similar to Chohlas-Wood et al. (2024, p. 14), we rewrite the objective function in order to linearise the optimisation problem by introducing an additional slack variable $w$, yielding a mixed integer linear program (MILP) instead of a MIQP.[10] Combining all elements, the policy-maker's full optimisation problem reads:

[10] This approach allows us as well to transform the optimisation problem in Equation 10 with the absolute prediction disparity as objective function from a MIQP into a MIQCP.

$$\max_{\pi} \hat{p}_0 \langle \hat{\mathbf{\Gamma}}_{1,0} - \hat{\mathbf{\Gamma}}_{0,0}, \mathbf{z}_0 \rangle + \hat{p}_1 \langle \hat{\mathbf{\Gamma}}_{1,1} - \hat{\mathbf{\Gamma}}_{0,1}, \mathbf{z}_1 \rangle - \lambda w \quad \text{subject to} \tag{18}$$

$$z_{s,i} = \pi(X_i, s), \quad 1 \leq i \leq n, \tag{A}$$

$$-w \leq \left[ \frac{1}{n} \sum_{i=1}^{n} \frac{(1 - S_i)}{\hat{p}_0} z_{0,i} - \frac{1}{n} \sum_{i=1}^{n} \frac{S_i}{\hat{p}_1} z_{1,i} \right] \leq w \tag{B}$$

$$w \geq 0 \tag{C}$$

$$\pi \in \Pi \tag{D}$$

Once again, the linear constraint in (A) represents the additional $n$ binary decision variables introduced for the MIP representation. Constraints (B) and (C) allow us to linearise the objective function $\hat{U}(\pi)$ and hence convert the optimisation problem into a MILP. Lastly, constraint (D) again requires the policy function to belong to the respective function class. The pseudocode for both, the regularised as well as the FPT approach can be found in Appendix A.

# 4 Empirical Application

In this section we describe the application of the proposed FPT and regularised method to data from the National Study of Learning Mindsets. As in Viviano and Bradic (2024, p. 739), the goal is to define a policy that assigns students to an educational programme while ensuring fairness between genders. In Section 4.1 we describe the data and treatment. Section 4.2 discusses the design of the experiment and estimation methodology. Section 4.3 presents, analyses, and compares the results of the different approaches. Finally, Section 4.4 describes the challenges faced during the estimation and optimisation processes.

## 4.1 Data

The National Study of Learning Mindsets is an RCT that studied the effect of an intervention – an online programme designed to equip students with a growth learning mindset – on their subsequent academic performance (Student Experience Research Network, 2015).[11] Following Dweck and Yeager (2019), a growth mindset is defined as "the belief that human capacities are not fixed but can be developed over time" (Abstract).

For the present analysis, we follow Athey and Wager (2019, p. 1) and use a simulated dataset derived from a model fit to the original data.[12] The dataset comprises $n = 10\,391$ observations, each of which corresponds to a student from one of 76 participating schools. The analysed outcome $Y_i$ is a continuous measure of student $i$'s post-treatment academic performance. The treatment $D_i$ is binary, and indicates whether student $i$ participated in the programme ($D_i = 1$) or not ($D_i = 0$). The sensitive attribute $S_i$ captures student $i$'s self-identified gender, where $S_i = 1$ denotes female students and $S_i = 0$ male students. Additionally, there are 9 covariates measured on the student and on the school level that are described in Table 1.

[11]For more information about the study, see e.g., (Gopalan & Tipton, 2018; Yeager et al., 2016).
[12]The data can be found in the following Github repository.

| Variable | Description |
|---|---|
| $S3$ | The student's expectations of future success (continuous) |
| $C3$ | First member of the family to go to college (binary) |
| $X0$ | Urbanicity of the school (categorical) |
| $X1$ | School-level mean of students' fixed mindsets (continuous) |
| $X2$ | School-level achievement (continuous) |
| $X3$ | School-level share of ethnic minorities (continuous) |
| $X4$ | School-level share of students who are from families with incomes below the federal poverty line (continuous) |
| $X5$ | School size as total number of students (continuous) |

Table 1: Covariates measured in National Study of Learning Mindsets

*Source:* Athey and Wager (2019, p. 2)

## 4.2 Estimation Methodology

### 4.2.1 Propensity Score and Conditional Mean

Even though the original study was conceived of and conducted as an RCT, there are non-random selection effects present in the data (Athey & Wager, 2019, p. 2).[13] Therefore, we treated the data as coming from an observational study, and as such, the propensity score is unknown and needs to be estimated from the data (Rosenbaum & Rubin, 1983, p. 43). We followed the approach of Viviano and Bradic (2024) in so far as we used a $\ell^1$-penalised logistic regression and cross-fitting with five folds for the estimation of both, the propensity score and the conditional mean function (Appendix C.1). Moreover, we made use of the cross-validation feature of the `cv.glmnet` method contained in the `glm` package. In the propensity score regression, we conditioned on all the covariates except for the treatment, and used one-hot encoding to expand out the two categorical variables – the student's self-identified ethnicity and geographical location. Figure 1 shows the distribution of the estimated propensity score for treated and untreated individuals. It can be observed that the distributions of the two subgroups are overlapping and that the (conditional) probability of treatment is never equal to 0 or 1, confirming the positivity and overlap

[13]As they note further, with the non-random sampling, generalisations of the treatment effect beyond the participating schools are hard to argue for due to potentially unobserved school-level variables that could have a relevant effect on the outcome (Athey & Wager, 2019, p. 3). This problem is addressed by (1) using the doubly robust estimator and (2) refraining from generalisation beyond the participating schools as the main focus here lies with finding a suitable targeting rule, and not causal effects.

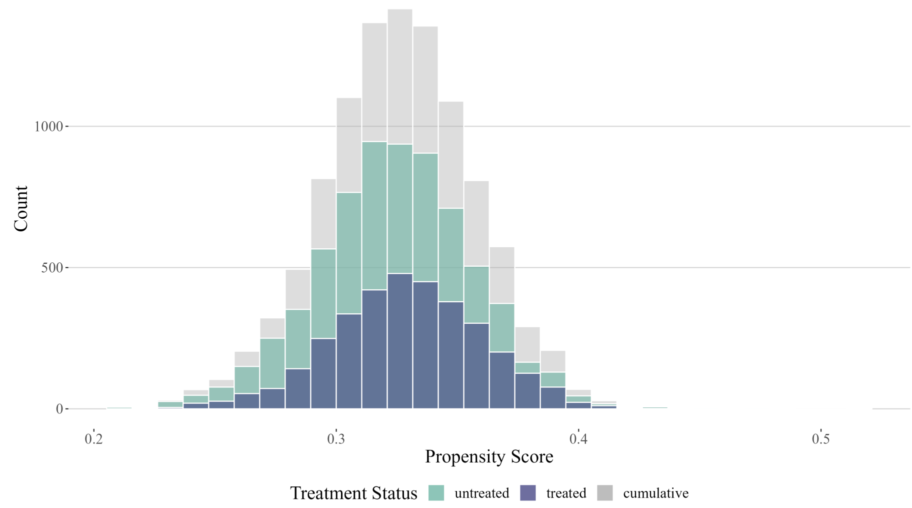


Figure 1: Comparison of the distribution of the estimated propensity score

assumptions (Zhu et al., 2021, p. 1472). For the conditional mean regression, we used the same covariates including, however, the treatment.

### 4.2.2 EWM and FPT Algorithms

For the allocation of the policies, we resorted to the policy function class used in Viviano and Bradic (2024, p. 739). This conforms to the decision set of the linear eligibility score defined in Kitagawa and Tetenov (2018, p. 598), where an individual is assigned to treatment if her eligibility score, i.e., a linear function of the individual's characteristics, is positive.

$$\Pi = \{\pi(X, S) = \mathbf{1}\{\beta_0 + \beta_1 S + \boldsymbol{\beta}^T X \geq 0\}\} \tag{19}$$

Besides the sensitive attribute, we used 6 covariates in $X$, namely (1) whether student $i$ is the first of her family to go to college; (2) the school-level mean of the students' mindset prior to the intervention; (3) school achievement level; (4) the school's minority composition; (5) the school's poverty concentration; and (6) school size. Moreover, we restricted the share of students that can receive the intervention to at most one-third.[14] The rationale behind this is to impose a budget constraint where we assume equal spending per student (Chohlas-Wood et al., 2024, p. 11;

[14]This will add an additional constraint on the binary decision variables $z_{s,i}$ to the respective optimisation problems in Equations 10 and Equation 18 in the form of $\sum_{s\in S}\sum_{i:S_i=s} z_{s,i} \leq \frac{1}{3}n$.

Nilforoshan et al., 2022, p. 2).

We analyse and contrast: (1) the results of the FPT algorithm with the fairness definitions presented in Section 3.2.3 – prediction disparity, absolute prediction disparity, and counterfactual envy; (2) the regularised approach presented in Section 3.3 with three different penalty terms $\lambda \in \{0.004,\ 0.010,\ 0.100\}$; and (3) the EWM methodology (Athey & Wager, 2021; Kitagawa & Tetenov, 2018). In selecting the penalty terms we first chose the $\lambda$ Chohlas-Wood et al. (2024) worked with, and subsequently let it increase, corresponding to more emphasis on equal assignment probabilities. In contrast, Kock and Preinerstorfer (2024) propose a data-driven method to select the optimal lambda $\lambda^*$ from a finite parameter space $\Lambda$, where $0 \in \Lambda$. Since adding a penalty term to the optimisation problem leads to lower overall welfare, they assume "a maximal budget the DM [decision-maker] can afford to spend in penalising for [inter-group] discrimination" (p. 27). The decision-maker then chooses $\lambda^*$ as the largest value for lambda (corresponding to a higher emphasis on fairness), given that it does not "overspend" on the fairness budget.[15] For the EWM approach, we adopted the three policy function classes proposed by Viviano and Bradic (2024, p. 739), where the first one comprises the policy function class defined in Equation 19. The second constrains the coefficient of the sensitive attribute $\beta_1$ to equal zero, reminiscent of a «fairness through unawareness» approach. Finally, in comparison to the second, the third class additionally requires that the average welfare of females must be at least as large as for males, estimated using the doubly robust scores. Formally, we have:

$$\begin{aligned}
\Pi_1 &= \{\pi(X,S) = \mathbf{1}\{\beta_0 + \beta_1 S + \boldsymbol{\beta}^T X \geq 0\}\} \\
\Pi_2 &= \{\pi(X) = \mathbf{1}\{\beta_0 + \boldsymbol{\beta}^T X \geq 0\}\} \\
\Pi_3 &= \{\pi(X) = \mathbf{1}\{\beta_0 + \boldsymbol{\beta}^T X \geq 0\}\} \\
&\quad \cap\ \mathbb{E}_n[(\Gamma_{1,i} - \Gamma_{0,i})\pi(X_i)|S=1] \geq \mathbb{E}_n[(\Gamma_{1,i} - \Gamma_{0,i})\pi(X_i)|S=0]
\end{aligned} \tag{20}$$

Due to the computational complexity of the FPT algorithm, we had to modify the dataset in two regards. Viviano and Bradic (2024) note that when including

[15] For additional information on the estimation procedure we refer the interested reader to Kock and Preinerstorfer (2024).

continuous variables in the maximum score function, the optimisation problem becomes "NP-hard in the worst-case scenario, hence, infeasible for large samples" (p. 737). Consequently, in a first step, we converted the real-valued covariates into binary variables, by assigning them the value of 1 if $X_i > \hat{\mu}_X$, where $\hat{\mu}_X$ corresponds to the respective sample mean of $X$, and 0 otherwise. Since the resulting data frame was still too large for RStudio and our hardware to process, we further reduced the size of the dataset to a random sample of $n = 500$ observations. However, we took the sample only after the estimation of the nuisance functions, such that we could still take advantage of the larger sample size for the estimation of the latter. Table 2 compares the summary statistics of the original dataset with the sample. Moreover, Viviano and Bradic (2024) show that with an early termination strategy, and consequently a certain optimality gap, the algorithm still achieves an informative result for the estimated unfairness (Proposition 2.7).

| | D | | S | | Propensity Score | | Conditional Mean Treated | | Conditional Mean Untreated | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Sample | Data | Sample | Data | Sample | Data | Sample | Data | Sample | Data |
| Min | 0.000 | 0.000 | 0.000 | 0.000 | 0.223 | 0.207 | 0.181 | 0.177 | 0.120 | 0.117 |
| 1st Q. | 0.000 | 0.000 | 0.000 | 0.000 | 0.306 | 0.305 | 0.475 | 0.473 | 0.418 | 0.415 |
| Median | 0.000 | 0.000 | 0.000 | 0.000 | 0.327 | 0.326 | 0.519 | 0.520 | 0.462 | 0.461 |
| Mean | 0.304 | 0.326 | 0.492 | 0.490 | 0.326 | 0.326 | 0.517 | 0.516 | 0.459 | 0.459 |
| 3rd Q. | 1.000 | 1.000 | 1.000 | 1.000 | 0.348 | 0.347 | 0.564 | 0.564 | 0.505 | 0.507 |
| Max | 1.000 | 1.000 | 1.000 | 1.000 | 0.418 | 0.512 | 0.699 | 0.769 | 0.644 | 0.708 |

Table 2: Comparison of the summary statistics of the original dataset and the sample taken with $n = 500$

## 4.3 Results

We carried out all computations and analyses with R in the RStudio environment. To solve the optimisation problems, we used the GUROBI Optimiser's R API, which is freely available for academic use. All code files are available in Appendix B.

Starting with the analysis of the EWM method, it is observable that the optimisation's objective value – which, of course, is relative – decreases as expected with each additional constraint, from 18.573 for the first policy function class to 16.438 and 15.168 for the second and third class, respectively. In the style of Vi-

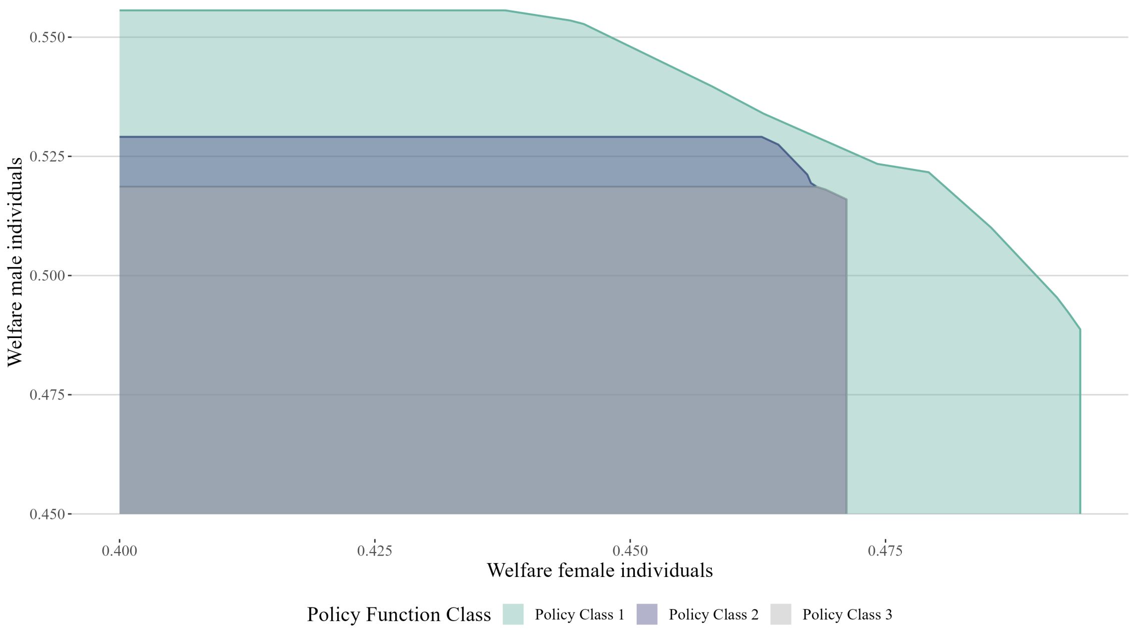


Figure 2: The discretised Pareto frontier of the three different policy function classes

viano and Bradic (2024, p. 740), Figure 2 shows the discretised Pareto frontier of the three decision sets. To get a finer approximation of the frontier, we increased the grid size of the equally spaced weights $\alpha_j \in (0, 1)$, where $j \in \{1, \dots, N\}$ to $N = 100$. It becomes apparent that fairness constraints on the policy function class yield Pareto-dominated outcomes. This in turn justifies the proposed methodology that first imposes Pareto optimality in the least constrained class and from there selects the least unfair allocation.

Table 3 reports the welfare effects of the different methods for female and male students, the importance weight associated with the female group (for the regularised method, the column reflects the respective value of the penalty term $\lambda$), the run time of the optimisation process, and the remaining optimality gap. It is observable that the EWM approach achieves its global optimum, whereas the FPT and regularised methodologies remain with rather large optimality gaps – despite rather extensive time limits of $15\,000s$ for each optimisation. Only minimising prediction disparity results in an optimality gap lower than $0.01\%$, which is why the optimisation stopped earlier. The welfare columns report the welfare improvement due to the treatment plus the baseline value for the two respective groups. The fact that the three methods counterfactual envy, absolute prediction disparity, and the welfare maximisation of the first policy class lead to equivalent welfare outcomes despite differing weights for the female group is a result of the discreteness of the Pareto frontier (Viviano & Bradic, 2024, p. 740), and of rounding to three decimal places.

The analysis of the welfare resulting from the EWM approach shows that the change from $\Pi_1$ to $\Pi_2$ entails a noticeable increase in welfare for the male group, while the female group experiences a strong decrease. Further tweaking the regime to $\Pi_3$ turns the movement in the opposite direction. However, the initial welfare levels for both groups remain unattainable.

The results of the FPT algorithms reveal that only with minimising prediction disparity, the welfare of the female group is larger than the welfare of the male group. With the rather large optimality gaps of the counterfactual envy and absolute prediction disparity metrics in mind, one could hypothesise that solutions closer to the global optimum would exhibit similar results. Concerning the importance weights assigned to the female group, the data reflect that each metric independently detects and applies the weight that minimises the respective unfairness measure without recourse to prior specification.

| Method | Welfare female | Welfare male | Weight | Time ($s$) | Gap (%) |
|---|---|---|---|---|---|
| Counterfactual Envy | 0.479 | 0.522 | 0.564 | 15 000 | 227.31 |
| Prediction Disparity | 0.494 | 0.489 | 0.821 | 12 088 | 0.01 |
| Prediction Disparity Abs | 0.479 | 0.522 | 0.522 | 15 000 | 83.03 |
| Welfare Max. $\Pi_1$ | 0.479 | 0.522 | 0.492 | 727 | 0.00 |
| Welfare Max. $\Pi_2$ | 0.463 | 0.529 | 0.492 | 2 | 0.00 |
| Welfare Max. $\Pi_3$ | 0.469 | 0.518 | 0.492 | 3 | 0.00 |
| Regularised 1 | 0.478 | 0.520 | 0.004 | 15 000 | 37.93 |
| Regularised 2 | 0.464 | 0.528 | 0.010 | 15 000 | 54.00 |
| Regularised 3 | 0.466 | 0.521 | 0.100 | 15 000 | 67.35 |

Table 3: Comparison of the generated welfare from the different methods

The findings of the regularised approach with different penalty terms demonstrate comparable performance to the EWM method – yet, with remaining optimality gaps and much longer computation times. However, they still reveal interesting insights with regard to the share of treated individuals over the two groups. Table 4 shows that an increasing penalty term on the differences in treatment probabilities eventually forces them to coincide. Moreover, the regularisation exclusively favours the male group since they benefit from higher treatment probabilities.

| | Penalty Term $\lambda$ | | |
|---|---|---|---|
| | 0.004 | 0.010 | 0.100 |
| Male | 0.25 | 0.34 | 0.33 |
| Female | 0.39 | 0.30 | 0.33 |

Table 4: Proportion of individuals treated under the Regularised approach with different penalty terms

Figure 3 compares the unfairness levels of the different methods, evaluated with the (absolute) differences in treatment assignment probabilities between groups, i.e., prediction disparity, as the unfairness measure. Specifically, we observe two features: (1) the FPT algorithm yields lower or equal (in the case of absolute prediction disparity) unfairness compared to other methods that do not constrain the decision space, and (2) the methods that achieve lower unfairness than the FPT methodology use, however, Pareto-dominated policies. This latter observation is best exemplified by the «Regularised 3» method with absolute prediction disparity as unfairness measure. In Figure 3, its unfairness level is hardly visible since there is little difference in the treatment assignment probabilities – as can be seen in Table 4, and as we would expect from a method that penalises this difference. However, the method yields Pareto-dominated policies and lower welfare as seen in Table 3. Similar reasoning applies to the other methods with low unfairness.

## 4.4 Challenges

While the GUROBI solver could easily find incumbent solutions to the optimisation problem with the EWM and regularised methods, it proved very difficult – even impossible – to solve the same problem but with the FPT methodology (multiple tries with different seeds and time limits of up to $20\,000s$ each could not achieve to find any feasible incumbent solution to the problem). For this, we see multiple reasons: First, a large part must be due to the increased computational complexity of the MIQP compared to the MILP. Second, the hardware on which the optimisation procedure runs as well as the exact order of the decision variables and constraints can play an important role, a phenomenon called "Performance Variability" (Miltenberger, 2024). Third, since even small changes such as the ordering of constraints


Figure 3: Comparison of the unfairness levels across the different methodologies

matter, one could hypothesise that minor discrepancies can cause rather large disturbances as well. The fact that R – and other software used for numerical analysis – can produce different solutions with discrepancies of magnitude of up to $1e-5$ to algebraically identical expressions might further add to the problem.

We undertook several approaches while trying to find an adequate solution to the problem outlined above. First, we rearranged the constraints and the decision variables which, however, did not produce noticeable changes nor improvements to the solving procedure. Second, we modified various of GUROBI's parameters that influence the optimisation procedure. Namely, we tried (1) the *Heuristics* parameter to increase the time spent on using heuristics to find a solution, (2) the *MIP Focus* parameter to set the focus on finding feasible solutions instead of focusing on optimising the objective bound, and (3) the *Improve Start Time* parameter to change strategies during the optimisation procedure, i.e., switch to using heuristics after a certain amount of time has passed or a certain number of nodes have been explored (GUROBI Optimization, 2024). However, manipulating those parameters did not help find feasible solutions either. Third, we experimented with the *Feasibility Tolerance* parameter for integer values. After increasing the parameter from $1e-9$ to $1e-5$, the solver immediately found feasible solutions that are, however, quite likely to be local but not global optima to the problem, therefore we considered this approach as an option of last resort only. Lastly, we made use of the *model$start* attribute of the GUROBI solver, which allows the researcher to provide a (potentially) feasible solution that the solver can use as a starting point. we computed this initial solution from the Pareto frontier's estimated values for the policies, beta coefficients, and optimal weighting factor. This indeed allowed the optimisation solver to find feasible solutions for the FPT algorithms.

# 5 Discussion

This paper aimed to implement and compare different methodologies for the estimation of treatment assignment rules, namely Viviano and Bradic's (2024) FPT algorithm, a regularised approach inspired by Chohlas-Wood et al. (2024) and Kock and Preinerstorfer (2024), and empirical welfare maximisation such as in Kitagawa and Tetenov (2018). We considered a setting in which the policymaker has preferences for both, efficiency in the sense of maximising expected welfare as well as fairness between two demographic groups, and hence must attempt to balance the trade-off between the two competing objectives. Using data from the National Study of Learning Mindsets, we showed that the FPT method, which first singles out Pareto-efficient policies and from there selects the least unfair allocation, does indeed lead to outcomes that are more efficient in the sense of Pareto optimality and «fairer» according to common definitions. The results underline the merit of a more consequentialist approach to policy learning, based on the outcomes and effects of policies, instead of an axiomatic, process-oriented approach based on formal fairness criteria. Nevertheless, the consequentialist approach comes with its own limitations and ample room for further research exists.

For instance, the clear specification of an objective function remains a major challenge, which involves two sub-problems: (1) precisely measuring the outcome of interest and accordingly finding the «correct» target to train the algorithm for prediction (see e.g., Obermeyer et al. (2019) who show the demographic bias in an algorithm that should predict health status but uses health care costs as target variable); and (2) eliciting policymakers' or groups' preferences in balancing competing fairness and efficiency objectives and formalising them in a utility function. For the latter, see Chohlas-Wood et al. (2024) and Koenecke et al. (2023) who use various survey and assessment techniques to infer preferences, even without decision-makers explicitly stating them.

Along with our analysis, we implicitly took the stable unit treatment value assumption (SUTVA) as given, which is common in the policy learning literature. However, this assumption may not be appropriate in all settings. For instance, in the context of job training programmes or disease control, an individual's potential outcomes may change depending on other individuals' treatment assignments. In the presence of spillover effects, estimating treatment effects becomes increasingly complex and requires sophisticated analytical methods. For instance, see Forastiere et al. (2021) who estimate treatment effects by splitting the intervention's causal and spillover effects and show the introduced bias when unduly assuming SUTVA. In policy learning, one of the exceptions is Viviano (2024) who explores empirical welfare maximisation with network interference, however, without specific considerations regarding fairness.

# References

Ahrens, A., Stampi-Bombelli, A., Kurer, S., & Hangartner, D. (2024). *Optimal multi-action treatment allocation: A two-phase field experiment to boost immigrant naturalization*. arXiv: 2305.00545 [econ.GN].

Athey, S., & Imbens, G. (2016). Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, *113*(27), 7353–7360. https://doi.org/10.1073/pnas.1510489113

Athey, S., & Wager, S. (2019). *Estimating treatment effects with causal forests: An application*. arXiv: 1902.07409 [stat.ME].

Athey, S., & Wager, S. (2021). Policy learning with observational data. *Econometrica*, *89*(1), 133–161. https://doi.org/https://doi.org/10.3982/ECTA15732

Bonanno, G. (2017). Decision making. In *Expected utility theory* (pp. 73–100). https://faculty.econ.ucdavis.edu/faculty/bonanno/PDF/DM_book.pdf

Chernozhukov, V., Escanciano, J. C., Ichimura, H., Newey, W. K., & Robins, J. M. (2022). Locally robust semiparametric estimation. *Econometrica*, *90*(4), 1501–1535. https://doi.org/https://doi.org/10.3982/ECTA16294

Chernozhukov, V., Newey, W. K., & Robins, J. (2018). *Double/de-biased machine learning using regularized riesz representers* (cemmap working paper No. CWP15/18). London, Centre for Microdata Methods; Practice (cemmap). https://doi.org/10.1920/wp.cem.2018.1518

Chohlas-Wood, A., Coots, M., Zhu, H., Brunskill, E., & Goel, S. (2024). *Learning to be fair: A consequentialist approach to equitable decision-making*. arXiv: 2109.08792 [cs.LG].

Chouldechova, A., & Roth, A. (2018). *The frontiers of fairness in machine learning*. arXiv: 1810.08810 [cs.LG].

Corbett-Davies, S., Gaebler, J. D., Nilforoshan, H., Shroff, R., & Goel, S. (2023). *The measure and mismeasure of fairness*. arXiv: 1808.00023 [cs.CY].

Cowgill, B., & Tucker, C. E. (2020). Algorithmic fairness and economics. *Columbia Business School Research Paper*. https://ssrn.com/abstract=3361280

Dweck, C. S., & Yeager, D. S. (2019). Mindsets: A view from two eras [PMID: 30707853]. *Perspectives on Psychological Science*, *14*(3), 481–496. https://doi.org/10.1177/1745691618804166

Fang, E. X., Wang, Z., & Wang, L. (2023). Fairness-oriented learning for optimal individualized treatment rules. *Journal of the American Statistical Association*, *118*(543), 1733–1746. https://doi.org/10.1080/01621459.2021.2008402

Florios, K., & Skouras, S. (2008). Exact computation of max weighted score estimators. *Journal of Econometrics*, *146*(1), 86–91. https://doi.org/https://doi.org/10.1016/j.jeconom.2008.05.018

Forastiere, L., Airoldi, E. M., & Mealli, F. (2021). Identification and estimation of treatment and interference effects in observational studies on networks. *Journal of the American Statistical Association*, *116*(534), 901–918. https://doi.org/10.1080/01621459.2020.1768100

Frölich, M. (2008). Statistical treatment choice. *Journal of the American Statistical Association*, *103*(482), 547–558. https://doi.org/10.1198/016214507000000572

Fuster, A., Goldsmith-Pinkham, P., Ramadorai, T., & Walther, A. (2022). Predictably unequal? the effects of machine learning on credit markets. *The Journal of Finance*, *77*(1), 5–47. https://doi.org/https://doi.org/10.1111/jofi.13090

Gopalan, M., & Tipton, E. (2018). Is the national study of learning mindsets nationally-representative? *PsyArXiv*. https://doi.org/10.31234/osf.io/dvmr7

Gupta, S., Jalan, A., Ranade, G., Yang, H., & Zhuang, S. (2020). Too many fairness metrics: Is there a solution? equity across demographic groups for the facility location problem. https://ssrn.com/abstract=3554829

GUROBI Optimization. (2024). *Gurobi optimizer reference manual*. Retrieved February 8, 2024, from https://www.gurobi.com/documentation/current/refman/refman.html

Hardt, M., Price, E., & Srebro, N. (2016). *Equality of opportunity in supervised learning*. arXiv: 1610.02413 `[cs.LG]`.

Hirano, K., & Porter, J. R. (2009). Asymptotics for statistical treatment rules. *Econometrica*, *77*(5), 1683–1701. http://www.jstor.org/stable/25621374

Ida, T., Ishihara, T., Ito, K., Kido, D., Kitagawa, T., Sakaguchi, S., & Sasaki, S. (2022, September). *Choosing who chooses: Selection-driven targeting in energy rebate programs* (Working Paper No. 30469). National Bureau of Economic Research. https://doi.org/10.3386/w30469

Imbens, G. W., & Rubin, D. B. (2015). Causality: The basic framework. In *Causal inference for statistics, social, and biomedical sciences: An introduction* (pp. 3–22). Cambridge University Press.

Kitagawa, T., & Tetenov, A. (2018). Who should be treated? empirical welfare maximization methods for treatment choice. *Econometrica*, *86*(2), 591–616. https://doi.org/https://doi.org/10.3982/ECTA13288

Kleinberg, J., Ludwig, J., Mullainathan, S., & Rambachan, A. (2018). Algorithmic fairness. *AEA Papers and Proceedings*, *108*, 22–27. https://doi.org/10.1257/pandp.20181018

Kock, A. B., & Preinerstorfer, D. (2024). *Regularizing discrimination in optimal policy learning with distributional targets*. arXiv: 2401.17909 [econ.EM].

Koenecke, A., Giannella, E., Willer, R., & Goel, S. (2023). *Popular support for balancing equity and efficiency in resource allocation: A case study in online advertising to increase welfare program awareness*. arXiv: 2304.08530 [cs.CY].

Kusner, M. J., Loftus, J., Russell, C., & Silva, R. (2017). Counterfactual fairness. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf

Lattimore, T., & Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press. https://doi.org/https://doi.org/10.1017/9781108571401

Manski, C. F. (2004). Statistical treatment rules for heterogeneous populations. *Econometrica*, *72*(4), 1221–1246. http://www.jstor.org/stable/3598783

Miltenberger, M. (2024). *Why does gurobi perform differently on different machines?* Retrieved February 8, 2024, from https://support.gurobi.com/hc/en-us/

articles/360045849232-Why-does-Gurobi-perform-differently-on-different-machines

Nabi, R., Malinsky, D., & Shpitser, I. (2019). Learning optimal fair policies. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning: Vol. 97* (pp. 4674–4682). PMLR. https://proceedings.mlr.press/v97/nabi19a.html

Nilforoshan, H., Gaebler, J. D., Shroff, R., & Goel, S. (2022). Causal conceptions of fairness and their consequences. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th international conference on machine learning: Vol. 162.* (pp. 16848–16887). PMLR. https://proceedings.mlr.press/v162/nilforoshan22a.html

Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, *366*(6464), 447–453. https://doi.org/10.1126/science.aax2342

Rambachan, A., Kleinberg, J., Ludwig, J., & Mullainathan, S. (2020). An economic perspective on algorithmic fairness. *AEA Papers and Proceedings*, *110*, 91–95. https://doi.org/10.1257/pandp.20201036

Rambachan, A., Kleinberg, J., Mullainathan, S., & Ludwig, J. (2020, May). *An economic approach to regulating algorithms* (Working Paper No. 27111). National Bureau of Economic Research. https://doi.org/10.3386/w27111

Robins, J. M., & Rotnitzky, A. (1995). Semiparametric efficiency in multivariate regression models with missing data. *Journal of the American Statistical Association*, *90*(429), 122–129. https://doi.org/https://doi.org/10.2307/2291135

Robins, J. M., Rotnitzky, A., & Zhao, L. P. (1994). Estimation of regression coefficients when some regressors are not always observed. *Journal of the American Statistical Association*, *89*(427), 846–866. https://doi.org/https://doi.org/10.2307/2290910

Rosenbaum, P. R., & Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, *70*(1), 41–55. https://doi.org/10.2307/2335942

Student Experience Research Network. (2015). *National study of learning mindsets*. https://studentexperiencenetwork.org/national-mindset-study/

Tetenov, A. (2012). Statistical treatment choice based on asymmetric minimax regret criteria. *Journal of Econometrics*, *166*(1), 157–165. https://doi.org/https://doi.org/10.1016/j.jeconom.2011.06.013

Varian, H. R. (1976). Two problems in the theory of fairness. *Journal of Public Economics*, *5*(3), 249–260. https://doi.org/https://doi.org/10.1016/0047-2727(76)90018-9

Verma, S., & Rubin, J. (2018). Fairness definitions explained. *Proceedings of the International Workshop on Software Fairness*, 1–7. https://doi.org/10.1145/3194770.3194776

Viviano, D., & Bradic, J. (2024). Fair policy targeting. *Journal of the American Statistical Association*, *119*(545), 730–743. https://doi.org/10.1080/01621459.2022.2142591

Viviano, D. (2024). *Policy targeting under network interference*. arXiv: 1906.10258 [econ.EM].

Yeager, D. S., Hulleman, C. S., Hinojosa, C., Lee, H. Y., O'Brien, J., Romero, C., Paunesku, D., Schneider, B., Flint, K., Roberts, A., Trott, J., Greene, D., Walton, G. M., & Dweck, C. S. (2016). Using design thinking to improve psychological interventions: The case of the growth mindset during the transition to high school. *Journal of educational psychology*, *108*, 374–391. https://doi.org/10.1037/EDU0000098

Zhou, Z., Athey, S., & Wager, S. (2023). Offline multi-action policy learning: Generalization and optimization. *Operations Research*, *71*(1), 148–183. https://doi.org/10.1287/opre.2022.2271

Zhu, Y., Hubbard, R. A., Chubak, J., Roy, J., & Mitra, N. (2021). Core concepts in pharmacoepidemiology: Violations of the positivity assumption in the causal analysis of observational data: Consequences and statistical approaches. *Pharmacoepidemiology and Drug Safety*, *30*(11), 1471–1485. https://doi.org/https://doi.org/10.1002/pds.5338

# A Algorithms

## A.1 Fair Policy Targeting

**Algorithm 1** Fair Policy Targeting Algorithm

1: **Input**: Outcome $Y_i$, covariates $X_i$, sensitive attribute $S_i$, treatment assignment $D_i$, estimated propensity score $\hat{e}_d$ conditional mean $\hat{m}_{d,s}$ and $\hat{p}_s$, discretised importance weights $\alpha$, capacity constraint $k$

2: Normalise the data.

3: Compute the doubly robust scores $\hat{\Gamma}_{d,s,i} = \frac{\mathbf{1}\{S_i=s\}}{\hat{p}_s}\left[\frac{\mathbf{1}\{D_i=d\}}{\hat{e}_d(X_i,S_i)}(Y_i - \hat{m}_{d,s}(X_i)) + \hat{m}_{d,s}(X_i)\right]$.

4: Estimate the largest empirical welfare $\bar{W}_{j,n} = \sup_{\pi\in\Pi}\left\{\alpha_j \hat{W}_0(\pi) + (1-\alpha_j)\hat{W}_1(\pi)\right\}$ for all $j$ with corresponding policy function class $\Pi = \{\pi(X,S) = \mathbf{1}\{\beta_0 + \beta_1 S + \boldsymbol{\beta}^T X \geq 0\}\}$ via:[16]

5: **for** $\alpha_j \in (\alpha_1, \dots, \alpha_N)$ **do**

6: $\quad \max_{z_{s,i},\beta_0,\beta_1,\boldsymbol{\beta}} \alpha_j \langle \hat{\mathbf{\Gamma}}_{1,0} - \hat{\mathbf{\Gamma}}_{0,0}, \mathbf{z}_0\rangle + (1-\alpha_j)\langle \hat{\mathbf{\Gamma}}_{1,1} - \hat{\mathbf{\Gamma}}_{0,1}, \mathbf{z}_1\rangle$

$$\begin{aligned}
\text{subject to} \quad & z_{s,i} = \pi(X_i, s) \in \{0,1\}, \quad 1 \leq i \leq n \\
& \tfrac{\beta_0+\beta_1 s+\boldsymbol{\beta}^T X_i}{|C_i|} < z_{s,i} \leq 1 + \tfrac{\beta_0+\beta_1 s+\boldsymbol{\beta}^T X_i}{|C_i|} \\
& C_i > \sup_{\{\beta_0,\beta_1,\boldsymbol{\beta}\}\in\mathcal{B}} |\boldsymbol{\beta}^T X_i| + |\beta_1| + |\beta_0| \\
& \textstyle\sum_{s\in S}\sum_{i:S_i=s} z_{s,i} \leq k
\end{aligned}$$

7: $\quad$ Save the objective value in a vector.

8: **end for**

9: Find the optimal solution $\hat{\pi}_\lambda$ to the optimisation problem in Equation 10 with the according fairness measure and including the capacity constraint.

10: **if** Counterfactual envy **then**

11: $\quad \hat{\mathcal{V}}(\pi) = \hat{\mathcal{A}}(0,1;\pi) + \hat{\mathcal{A}}(1,0;\pi)$

12: **else if** Prediction disparity **then**

13: $\quad \hat{\mathcal{V}}(\pi) = \frac{1}{n}\sum_{i=1}^n \frac{\pi(X_i)(1-S_i)}{(1-\hat{p}_1)} - \frac{1}{n}\sum_{i=1}^n \frac{\pi(X_i)S_i}{\hat{p}_1}$

14: **else if** Absolute prediction disparity **then**

15: $\quad \hat{\mathcal{V}}(\pi) = \left|\frac{1}{n}\sum_{i=1}^n \frac{\pi(X_i)(1-S_i)}{(1-\hat{p}_1)} - \frac{1}{n}\sum_{i=1}^n \frac{\pi(X_i)S_i}{\hat{p}_1}\right|$

16: **end if**

17: **Return:** Optimal policy function $\hat{\pi}_\lambda$, importance weight $\alpha_j$, objective value $\hat{\mathcal{V}}$, optimisation time, optimality gap

Pseudocode structure inspired by Zhou et al. (2023, p. 14) and Chohlas-Wood et al. (2024, p. 22)

[16] See Kitagawa and Tetenov (2018, Appendix C.1 and C.3) and Viviano and Bradic (2024, Appendix B.3)

## A.2 Regularised Approach

**Algorithm 2** Regularised Approach

1: **Input**: Outcome $Y_i$, covariates $X_i$, sensitive attribute $S_i$, treatment assignment $D_i$, estimated propensity score $\hat{e}_d$ conditional mean $\hat{m}_{d,s}$ and $\hat{p}_s$, penalty term $\lambda$, capacity constraint $k$

2: Normalise the data.

3: Compute the doubly robust scores $\hat{\Gamma}_{d,s,i} = \frac{\mathbf{1}\{S_i=s\}}{\hat{p}_s}\left[\frac{\mathbf{1}\{D_i=d\}}{\hat{e}_d(X_i,S_i)}(Y_i - \hat{m}_{d,s}(X_i)) + \hat{m}_{d,s}(X_i)\right].$

4: Find the optimal solution $\hat{\pi}$ to the optimisation problem in Equation 18 with respective importance weights $\alpha = \hat{p}_0, (1-\alpha) = \hat{p}_1$, penalty term $\lambda$, capacity constraint $k$, and with corresponding policy function class $\Pi = \{\pi(X,S) = \mathbf{1}\{\beta_0 + \beta_1 S + \boldsymbol{\beta}^T X \geq 0\}\}$ via:

5: $\max_{\pi} \hat{p}_0 \langle \hat{\boldsymbol{\Gamma}}_{1,0} - \hat{\boldsymbol{\Gamma}}_{0,0}, \mathbf{z}_0 \rangle + \hat{p}_1 \langle \hat{\boldsymbol{\Gamma}}_{1,1} - \hat{\boldsymbol{\Gamma}}_{0,1}, \mathbf{z}_1 \rangle - \lambda w$

$$
\begin{aligned}
\text{subject to} \quad & z_{s,i} = \pi(X_i, s) \in \{0,1\}, \quad 1 \leq i \leq n \\
& \tfrac{\beta_0+\beta_1 s+\boldsymbol{\beta}^T X_i}{|C_i|} < z_{s,i} \leq 1 + \tfrac{\beta_0+\beta_1 s+\boldsymbol{\beta}^T X_i}{|C_i|} \\
& C_i > \sup_{\{\beta_0,\beta_1,\boldsymbol{\beta}\}\in\mathcal{B}} |\boldsymbol{\beta}^T X| + |\beta_1| + |\beta_0| \\
& -w \leq \left[\tfrac{1}{n}\textstyle\sum_{i=1}^n \tfrac{(1-S_i)}{\hat{p}_0} z_{0,i} - \tfrac{1}{n}\sum_{i=1}^n \tfrac{S_i}{\hat{p}_1} z_{1,i}\right] \leq w, \quad w \geq 0 \\
& \textstyle\sum_{s\in S}\sum_{i:S_i=s} z_{s,i} \leq k
\end{aligned}
$$

6: **Return:** Optimal policy function $\hat{\pi}$, importance weight $\alpha_j$, objective value $\hat{\mathcal{V}}$, optimisation time, optimality gap

Pseudocode structure inspired by Zhou et al. (2023, p. 14) and Chohlas-Wood et al. (2024, p. 22)

# B Code Files

## B.1 Base Estimation

```r
##############################################################

# Implementation: National Study of Learning Mindset

##############################################################

# load libraries
library(tidyverse)
library(data.table)
library(mltools)
library(caret)
library(glmnet)
library(extrafont)
font_import()

#### Data preparation
data = read_csv("./data/synthetic_data.csv")

# prepare data
data = data %>% mutate(S = ifelse(data$C2 == 2, 1, 0), # recode S (0,1)
                        D = Z,                        # treatment vector D
                        Y = 1 / (1+exp(-data$Y)))     # normalise Y (0,1)

# one hot encoding for categorical data
data$C1 = as_factor(data$C1)
data$XC = as_factor(data$XC)
data = one_hot(as.data.table(data))



#### Propensity score estimation

# function for the propensity score estimation
# df: data frame for the estimation
# D: the treatment vector
# K: the number of folds for cross-fitting
estimate_ps = function(df, D, K = 5) {

  # load required libraries
  library(caret)
  library(glmnet)

  # create the folds for the cross-fitting
  set.seed(777)
  folds = createFolds(D, k = K)

  # storage for the predictions
  ps = rep(0, nrow(df))

  # initiate for loop
  for (i in 1:K){

    # split data in training and test set
    test_indices = unlist(folds[i])
    train_indices = unlist(folds[-i])

    # remove D as the first column of df
    x_train = as.matrix(df[train_indices, -1])
    y_train = D[train_indices]
    x_test = as.matrix(df[test_indices, -1])

    # cross validation for lambda
    ps_reg = cv.glmnet(x_train, as.factor(y_train), family = "binomial")

    # make predictions using the test set and optimal lambda
```

```r
    ps[test_indices] = predict(ps_reg, s = ps_reg$lambda.min, newx = x_test, type = "response")

  }

  # return the ps
  return(ps)

}

# prepare the df for the propensity score estimation
df_ps = data %>% select(D, S, everything(), -Y, -C2, -schoolid, -Z)

# estimate and add the propensity score to df_ps
df_ps$ps = estimate_ps(df = df_ps, D = df_ps$D, K = 5)

# plot to check overlap between treated and untreated (Positivity Check)
ggplot(df_ps) +
  geom_histogram(aes(x = ps, fill = "darkgrey"), color = "white", alpha = 0.6, position = 'identity', show.legend = T) +
  geom_histogram(aes(x = ps, fill = as.factor(D)), colour = "white", alpha = 0.6, position = "identity") +
  scale_fill_manual(values = c("#69b3a2", "#404080", "darkgrey"),
                    labels = c("untreated", "treated", "cumulative"), name = "Treatment Status")+
  labs(x = "Propensity Score", y = "Count") +
  theme_hc() +
  theme(text = element_text(family = "Times New Roman"),
        axis.title = element_text(size = 18),
        axis.text = element_text(size = 14),
        legend.title = element_text(size = 18),
        legend.text = element_text(size = 14))

ggsave("positivity_check.png", plot = last_plot(), path = "./results/", height = 7, width = 12)

#### Conditional mean estimation

# function for the conditional mean estimation
# Y: outcome variable
# S: sensitive attribute
# df: data frame for the estimation
# K: number of folds for crossfitting
estimate_conditional_mean = function(Y, S, df_mean_reg, K = 5){

  # create folds for cross-fitting
  set.seed(777)
  folds = createFolds(Y, k = K)

  # create storage for estimations m_d_s
  m11_hat = m10_hat = m01_hat = m00_hat = rep(0, length(Y))

  # initiate the for loop over the folds
  for (i in 1:K){

    # split data in training and testing set
    train_indices = unlist(folds[-i])
    test_indices = unlist(folds[i])

    # main regression for the prediction
    mean_reg = cv.glmnet(as.matrix(df_mean_reg[train_indices, ]), Y[train_indices], family = "gaussian")
    lambda = mean_reg$lambda.min

    # m11_hat: test data (assuming D = 1, S = 1) and prediction
    df_m11 = cbind(1, 1, df_mean_reg[,-c(1,2)])
    m11_hat[test_indices] = predict(mean_reg, s = lambda,  newx = as.matrix(df_m11)[test_indices, ], type = "response")

    # m01_hat: test data (assuming D = 0, S = 1) and prediction
```

```
    df_m01 = cbind(0, 1, df_mean_reg[,-c(1,2)])
    m01_hat[test_indices] = predict(mean_reg, s = lambda, newx = as.matrix(df_m01)[test_indices, ], type = "response")

    # m10_hat: test data (assuming D = 1, S = 1) and prediction
    df_m10 = cbind(1, 0, df_mean_reg[,-c(1,2)])
    m10_hat[test_indices] = predict(mean_reg, s = lambda, newx = as.matrix(df_m10)[test_indices, ], type = "response")

    # m00_hat: test data (assuming D = 0, S = 0) and prediction
    df_m00 = cbind(0, 0, df_mean_reg[,-c(1,2)])
    m00_hat[test_indices] = predict(mean_reg, s = lambda, newx = as.matrix(df_m00)[test_indices, ], type = "response")

  }

  # compute the conditional means of treated and untreated individuals
  m1 = S*m11_hat + (1-S)*m10_hat
  m0 = S*m01_hat + (1-S)*m00_hat

  return(list(m1 = m1, m0 = m0, m11_hat = m11_hat, m01_hat = m01_hat, m10_hat = m10_hat, m00_hat = m00_hat))

}

# prepare the df for the conditional mean estimation
df_mean_reg = data %>% select(D, S, everything(), -Y, -C2, -Z)

# estimate the conditional means
elements = estimate_conditional_mean(Y = data$Y, S = data$S, df_mean_reg = df_mean_reg, K = 5)



#### Prepare final dataset to be used for algorithms
df = data %>% select(Y, D, S, C3, paste0("X", c(1:5))) %>%
  mutate(X1_D = ifelse(X1 > mean(X1), 1, 0),
         X2_D = ifelse(X2 > mean(X2), 1, 0),
         X3_D = ifelse(X3 > mean(X3), 1, 0),
         X4_D = ifelse(X4 > mean(X4), 1, 0),
         X5_D = ifelse(X5 > mean(X5), 1, 0),
         ps = df_ps$ps, m1 = elements$m1, m0 = elements$m0,
         m11_hat = elements$m11_hat,
         m10_hat = elements$m10_hat,
         m01_hat = elements$m01_hat,
         m00_hat = elements$m00_hat)

# take a random sample of n = 500 observations for the optimisation
set.seed(000)
df_sample = slice_sample(.data = df, n = 500)


# comparison between sample and full dataset for important variables
df_sample_comparison = df_sample %>% select(D, S, ps, m1, m0)
data_comparison = data %>% select(D, S) %>%
  mutate(ps = df_ps$ps, m1 = elements$m1, m0 = elements$m0)

# xtable(summary(df_sample_comparison))
# xtable(summary(data_comparison))
# save(list = c("df_sample"), file = "./results/implementation.RData")
```

## B.2 EWM and Pareto Frontier Estimation

```
#############################################################################

# Empirical Welfare Maximisation (EWM): Function

#############################################################################

# Y: outcome
# X: matrix of covariates
# D: treatment vector
# S: sensitive attribute vector
# ps: estimated propensity score
# m1: conditional mean of treated individuals
# m0: conditional mean of untreated individuals
# alpha: weight of the male welfare
# tolerance: slackness parameter (discreteness)
# capacity_constraint: maximal number of individuals to be treated
# funclass3: policy function class 3 that requires ATE(S==s_1) >= ATE(S==s_2)
# parity_sense: sense of the additional constraint above
# timelimit: maximum time spent on optimisation in seconds

EWM_estimation = function(Y, X, D, S, ps, m1, m0, alpha = mean(1-S),
                          tolerance = 1e-6, capacity_constraint, funclass3 = F,
                          parity_sense = ">=", timelimit = 5000){

  # load libraries
  library(Matrix)
  library(slam)
  library(gurobi)

  # compute the doubly robust score (drs)
  # note: one could also add treatment cost here by Y - m_d - treatment.cost
  G11_hat = (S/mean(S)) * ((D/ps)*(Y-m1)+m1) # treated females
  G01_hat = (S/mean(S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated females
  G10_hat = ((1-S)/mean(1-S)) * ((D/ps)*(Y-m1)+m1) # treated males
  G00_hat = ((1-S)/mean(1-S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated males

  # compute the difference of the drs as in W_s and full vector G for W_bar
  GS1 = G11_hat - G01_hat # female
  GS0 = G10_hat - G00_hat # male
  G = alpha*GS0 + (1-alpha)*GS1 # EWM: weight corresponding to prevalence in the sample

  # include additional column for intercept
  X = as.matrix(cbind(1, X))

  # normalise the values by dividing by the max value (if not already normalised)
  max_val = max(apply(X, 1, function(x) max(abs(x))))
  X = as.matrix(X/max_val)

  ## Facilitate and speed up computation by "omitting" identical rows
  # find and store all unique rows
  unique_rows = unique(X)

  # store the indexes of the rows which are identical
  index_unique = apply(X, 1, function(y) which(apply(unique_rows, 1, function(x) all(y == x))))

  # sum up the drs for individuals which have exactly the same covariate values (i.e. sum up by unique index)
  G_unique = sapply(c(1:nrow(unique_rows)), function(x) sum(G[which(x == index_unique)]))
  GS1_unique = sapply(c(1:nrow(unique_rows)), function(x) sum(GS1[which(x == index_unique)]))
  GS0_unique = sapply(c(1:nrow(unique_rows)), function(x) sum(GS0[which(x == index_unique)]))

  # preliminaries
  X_unique = as.matrix(unique_rows) # unique data frame
```

```r
n = nrow(X_unique) # number of unique observations
p = ncol(X_unique) # number of columns (including the one for beta0)

# number of times a certain observation is repeated in the data (required for capacity constraint)
n_index_unique = sapply(c(1:n), function(x) sum(index_unique == x))


## initialise the model
model  = list()

# sense of optimisation, maximise welfare
model$modelsense = "max"

# model objective: G_unique as coefficient vector
# (betas are not directly in the objective, thus coefficient vector is 0)
model$obj = c(G_unique, rep(0, p))

# set up the linear constraint matrix (rhs only constants; requirement of gurobi solver)
A = rbind(cbind(diag(1, nrow = n), -X_unique), # policies - betas <= 1
          cbind(diag(1, nrow = n), -X_unique), # policies - betas > 0
          c(n_index_unique, rep(0, p))) # capacity constraint

# the rhs of the constraints, with tolerance (1e-6) and max treated individuals equal to the capacity constraint
rhs = c(rep(1 - tolerance, n), rep(tolerance, n), capacity_constraint)

# direction of constraints
sense = c(rep("<=", n), rep(">", n), "<=")

# additional constraint if we choose policy function class 3
if (funclass3 == T){
  A = rbind(A, c(GS1_unique - GS0_unique, rep(0, p))) # drs as estimated treatment effect
  rhs = c(rhs, 0) # treatment_effect|S=1 - treatment_effect|S=0 >= 0
  sense = c(sense, parity_sense) # treatment effect of female >= male / female <= male
}

# combine all parts
model$A = A
model$rhs = rhs
model$sense = sense

# variable types, we have n binary variables (z) and the rest continuous (beta)
model$vtype= c(rep("B", n), rep("C", p))

# Put bounds on the parameter space, for z [0,1] and for the betas [-1,1] (assumption from paper)
model$ub= c(rep(1, n), rep(1, p))
model$lb= c(rep(0, n), rep(-1, p))

# set additional parameters for the optimisation
params = list(IntFeasTol = 1e-9, FeasibilityTol = 1e-9, TimeLimit = timelimit, # solver tolerance limits
              BarConvTol = exp(-2), # tolerance on the barrier between primal and dual solution
              Disconnected=0, # degree of exploitation of (independent) submodels
              Heuristics=0, # fraction of runtime spent on feasibility heuristics
              NodefileStart = 0.5) # max node memory before writing to hard drive

# start the optimisation
result= gurobi(model, params = params)

# extract the estimated beta_hats that determine the policy pi
beta = result$x[(n+1):(n+p)]

# extract the estimated policies pi_hats
policies = apply(X, 1, function(x) ifelse(x%*%beta > 0, 1, 0))
```

```r

  # extract the welfare
  welfare = result$objval

  return(list(welfare = welfare, policies = policies, beta = beta, time = result$
      runtime, gap = result$mipgap))

}



###############################################################################

# Estimation EWM (for comparison with Fair Policy Targeting algorithms)

###############################################################################

#### Prepare data and sample
library(dplyr)
load("./results/implementation.RData")

Y = df_sample$Y
D = df_sample$D
S = df_sample$S

m1 = df_sample$m1
m0 = df_sample$m0
ps = df_sample$ps

X = df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D)
capacity_constraint = floor(0.33*nrow(X))



#### EWM Estimation

# EWM Policy Class 1 (no constraint)
EWM_P1 = EWM_estimation(Y, X = cbind(S, X), D, S, ps, m1, m0, alpha = mean(1-S),
                        capacity_constraint, tolerance = 1e-6, funclass3 = F,
                            timelimit = 5000)

# EWM Policy Class 2 (fairness through unawareness)
EWM_P2 = EWM_estimation(Y, X, D, S, ps, m1, m0, alpha = mean(1-S), capacity_
    constraint,
                        tolerance = 1e-6, funclass3 = F, timelimit = 5000)

# EWM Policy Class 3 (additional constraint)
EWM_P3 = EWM_estimation(Y, X, D, S, ps, m1, m0, alpha = mean(1-S), capacity_
    constraint,
                        tolerance = 1e-6, funclass3 = T, parity_sense = ">=",
                            timelimit = 5000)


# save(list = c("EWM_P1", "EWM_P2", "EWM_P3"), file = "./results/EWM_estimation.
    RData")



###############################################################################

# Estimation of the Pareto Frontier (for Fair Policy Targeting algorithms)

###############################################################################

# additional set up (discretisation of pareto frontier)
N = floor(sqrt(nrow(X)))
alpha = seq(from = 0.05, to = 0.95, length.out = N)

# load libraries for parallelisation
library(foreach)
```

```
library(future)
library(doFuture)
plan(multisession)



# Estimation Pareto Frontier

# use "dofuture" to parallelise computation (individual EWM estimations are
   independent of each other)
pareto_frontier = foreach(i = alpha, .combine = rbind) %dofuture% {

  # EWM estimation for respective alpha
  result = EWM_estimation(Y, X = cbind(S, X), D, S, ps, m1, m0, alpha = i,
      tolerance = 1e-3,
                          capacity_constraint, funclass3 = F, timelimit = 1000)

  # collect results
  c(result[[2]], result[[3]], result[[1]])

}

n = nrow(X)
p = ncol(X)

# extract values
pareto_policies = pareto_frontier[,(1:n)]
pareto_beta = pareto_frontier[,(n+1):(n+p+2)]
pareto_W_bar = pareto_frontier[,(n+p+3)]

# save(list = c("pareto_policies", "pareto_beta", "pareto_W_bar"), file = "./
   results/pareto_frontier.RData")
```

## B.3 FPT Counterfactual Envy

```
##############################################################################

# Fair Policy Targeting (Counterfactual Envy): Function

##############################################################################

# Y: outcome
# X: matrix of covariates
# D: treatment vector
# S: sensitive attribute vector
# ps: estimated propensity score
# m1: conditional mean of treated individuals
# m0: conditional mean of untreated individuals
# m_d_s: conditional mean of individual with D=d, S=s
# alpha: weight of the male welfare
# tolerance: slackness parameter (discreteness)
# W_bar: the pareto frontier
# capacity_constraint: maximal number of individuals to be treated
# start: possible starting value for the optimisation
# timelimit: maximum time spent on optimisation in seconds

optimiseCounterfactualEnvy = function(Y, X, D, S, m0, m1, m11_hat, m01_hat, m10_hat, m00_hat, ps,
                                      alpha = seq(from = 0.05, to = 0.95, length.out = N), tolerance = 1e-6,
                                      W_bar, capacity_constraint, start = NA, timelimit = 10000){


  # compute the doubly robust score (drs)
  G11_hat = (S/mean(S)) * ((D/ps)*(Y-m1)+m1) # treated females
  G01_hat = (S/mean(S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated females
  G10_hat = ((1-S)/mean(1-S)) * ((D/ps)*(Y-m1)+m1) # treated males
  G00_hat = ((1-S)/mean(1-S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated males

  GS1 = G11_hat - G01_hat # female drs
  GS0 = G10_hat - G00_hat # male drs

  # data: intercept, sensitive attribute, covariates (case 2: average score, school rank)
  # note on second column: represents S with counterfactual covariates needed for V_pi(x,s) (x(s), s')
  X2_female = as.matrix(cbind(1, 1, X[,-1]))
  X2_male = as.matrix(cbind(1, 0, X[,-1]))
  XX = rbind(X2_female, X2_male)

  # normalise the data
  max_val = max(apply(XX, 1, function(x) sum(abs(x))))
  XX = XX/max_val

  # set parameters
  n = nrow(XX)
  p = ncol(XX)

  ## initialise the model
  model = list()

  # minimise unfairness (counterfactual envy in this case)
  model$modelsense = "min"

  # objective coefficients counterfactual envy; estimator A_hat but with omitted constants!
  A_hat_female = m10_hat*S/mean(S) - m00_hat*S/mean(S) - GS1
  A_hat_male = m11_hat*(1-S)/mean(1-S) - m01_hat*(1-S)/mean(1-S) - GS0

  # objective: the coefficients computed above, betas and alphas are not part of objective
  model$obj = c(A_hat_female, A_hat_male, rep(0, p+N))
```

```r

# set the linear constraint matrix
model$A = rbind(cbind(diag(1, nrow = n), -XX, matrix(0, nrow = n, ncol = N)), # policies - betas <= 1
                cbind(diag(1, nrow = n), -XX, matrix(0, nrow = n, ncol = N)), # policies - betas > 0
                c(S, (1-S), rep(0, p+N)), # capacity constraint (first female then male individuals as in objective)
                c(rep(0, n+p), rep(1, N))) # constraint on the u_j (C) >= 1

# set the rhs (with tolerance 1e-6)
model$rhs = c(rep(1-tolerance, n), rep(tolerance, n), capacity_constraint, 1)

# set the constraint directions
model$sense = c(rep("<=", n), # policies - betas <= 1
                rep(">", n), # policies - betas > 0
                "<=", # capacity constraint
                ">=") # constraint on the u_j >=1

# set up the quadratic constraints (B)
listnames = c(1:N)
model$quadcon = sapply(listnames, function(x) NULL)

# initiate a for loop to add the N=sqrt(n) constraints
for (i in 1:N){

  # initiate the Qc matrix
  model$quadcon[[i]]$Qc = matrix(0, nrow = n+p+N, ncol = n+p+N)
  # set coefficients in the column of the respective u_j
  model$quadcon[[i]]$Qc[1:(n/2), n+p+i] = (1-alpha[i])*GS1 # female weighted with alpha
  model$quadcon[[i]]$Qc[(n/2 + 1):n, n+p+i] = alpha[i]*GS0 # male weighted with 1-alpha

  # initiate q vector for the linear terms (optimal welfare W_bar)
  model$quadcon[[i]]$q = rep(0, n+p+N)
  # insert the respective welfare (- as bring to lhs)
  model$quadcon[[i]]$q[n+p+i] = -W_bar[i]

  # set the rhs
  model$quadcon[[i]]$rhs = -tolerance

  # direction of constraint
  model$quadcon[[i]]$sense = ">="

}

# set the variable type
model$vtype = c(rep("B", n), rep("C", p), rep("B", N))

# set bounds: for z_i and u_j [0,1], for betas [-1,1]
model$ub = rep(1, n+p+N)
model$lb = c(rep(0, n), rep(-1, p), rep(0, N))

if (is.na(start[1]) == F) model$start = start

# set a list of parameters
params = list(IntFeasTol = 1e-9, FeasibilityTol = 1e-9, TimeLimit = timelimit, # tolerance limits
              BarConvTol = exp(-2), # tolerance on the barrier between primal and dual solution
              Disconnected=0, # degree of exploitation of (independent) submodels
              Heuristics=0, # fraction of runtime spent on feasibility heuristics
              NodefileStart = 0.5) # max node memory before writing to hard drive

# results
result = gurobi(model, params = params)

# extract betas, weight, policies, objective value
beta = result$x[(n+1):(n+p)] # the betas for deciding on the policy
```

```r
  u = result$x[(n+p+2):length(result$x)] # which u_j is equal to 1
  weight = (1-alpha[u == 1]) # store the respective optimal weight alpha_j
  policies = apply(cbind(1, as.matrix(X)), 1, function(x) ifelse(x%*%beta > 0, 1, 0)) # the estimated policies
  objval = result$objval # the minimised unfairness (counterfactual envy)

  # return estimated betas, weight of the female group in the optimisation, policies and objective value
  return(list(beta = beta, weight_female = weight, policies = policies, objval = objval,
              result.x = result$x, time = result$runtime, gap = result$mipgap))

}

################################################################################

# Estimation FPT Counterfactual Envy

################################################################################

library(slam)
library(Matrix)
library(gurobi)
library(dplyr)

#### Preliminaries
load("./results/implementation.RData")
load("./results/pareto_frontier.RData")

Y = df_sample$Y
D = df_sample$D
S = df_sample$S

m1 = df_sample$m1
m0 = df_sample$m0
m11_hat = df_sample$m11_hat
m10_hat = df_sample$m10_hat
m01_hat = df_sample$m01_hat
m00_hat = df_sample$m00_hat
ps = df_sample$ps

X = df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D)
capacity_constraint = floor(0.33*nrow(X))

N = floor(sqrt(nrow(X)))
alpha = seq(from = 0.05, to = 0.95, length.out = N)

G11_hat = (S/mean(S)) * ((D/ps)*(Y-m1)+m1) # treated females
G01_hat = (S/mean(S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated females
G10_hat = ((1-S)/mean(1-S)) * ((D/ps)*(Y-m1)+m1) # treated males
G00_hat = ((1-S)/mean(1-S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated males

GS1 = G11_hat - G01_hat # female drs
GS0 = G10_hat - G00_hat # male drs


#### FPT Counterfactual Envy Estimation

## Build start vector with the help of the pareto frontier values to help find an initial solution in the optimisation
Xfem = as.matrix(cbind(1, 1, X))
Xmale = as.matrix(cbind(1, 0, X))

# compute the policies that female / male individuals would be prescribed
fem_policies = t(apply(pareto_beta, 1, function(x) sapply(Xfem%*%x, function(y) ifelse(y > 0, 1, 0))))
male_policies = t(apply(pareto_beta, 1, function(x) sapply(Xmale%*%x, function(y) ifelse(y > 0, 1, 0))))

# compute the welfare female / male individuals would achieve with their allocated
```

```
    policy
fem_welfare = apply(fem_policies, 1, function(x) sum(GS1*x))
male_welfare = apply(male_policies, 1, function(x) sum(GS0*x))

# objective for the start vector
objective_fem = apply(fem_policies, 1, function(x) sum(m10_hat*x*S)/mean(S) +  sum(m00_hat*(1 - x)*S)/mean(S)) -  male_welfare
objective_male = apply(male_policies, 1, function(x) sum(m11_hat*x*(1 - S))/(1 - mean(S)) +  sum(m01_hat*(1 - x)*(1 - S))/(1 - mean(S))) - fem_welfare
objective_start = objective_fem + objective_male

# compute the least unfair objective combination, as there might be multiple take the one closest to equal weighting
least_unfair = which(objective_start == min(objective_start))
least_unfair = least_unfair[which.min(abs(least_unfair - N/2))]

# combine elements to the start vector
start_envy = c(fem_policies[least_unfair,], male_policies[least_unfair,], pareto_beta[least_unfair,], ifelse(c(1:N) == least_unfair, 1, 0))

# optimise!
FPT_counterfactual_envy = optimiseCounterfactualEnvy(Y, X = cbind(S, X), D, S, m0, m1, m11_hat, m01_hat, m10_hat, m00_hat,
                                                     ps, alpha = alpha, tolerance = 1e-6, capacity_constraint,
                                                     W_bar = pareto_W_bar, start = start_envy, timelimit = 15000)


# save(list = c("FPT_counterfactual_envy"), file = "./results/fpt_counterfactual_envy.RData")
```

## B.4 FPT Prediction Disparity

```r
################################################################################

# Fair Policy Targeting (Prediction Disparity): Function

################################################################################

# Y: outcome
# X: matrix of covariates
# D: treatment vector
# S: sensitive attribute vector
# ps: estimated propensity score
# m1: conditional mean of treated individuals
# m0: conditional mean of untreated individuals
# alpha: weight of the male welfare
# tolerance: slackness parameter (discreteness)
# W_bar: the pareto frontier
# capacity_constraint: maximal number of individuals to be treated
# start: possible starting value for the optimisation
# timelimit: maximum time spent on optimisation in seconds

optimisePredictionDisparity = function(Y, X, D, S, m0, m1, ps, alpha = seq(from = 0.05, to = 0.95, length.out = N),
                                       tolerance = 1e-6, W_bar, capacity_constraint, start = NA, timelimit = 10000){

  # compute the doubly robust score (drs)
  G11_hat = (S/mean(S)) * ((D/ps)*(Y-m1)+m1) # treated females
  G01_hat = (S/mean(S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated females
  G10_hat = ((1-S)/mean(1-S)) * ((D/ps)*(Y-m1)+m1) # treated males
  G00_hat = ((1-S)/mean(1-S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated males

  GS1 = G11_hat - G01_hat # female drs
  GS0 = G10_hat - G00_hat # male drs

  # add intercept for beta0, and normalise
  X = as.matrix(cbind(1, X))
  max_val = max(apply(X, 1, function(x) sum(abs(x))))
  XX = X/max_val

  # set parameters
  n = nrow(XX)
  p = ncol(XX)

  ## initialise the model
  model = list()

  # sense of the optimisation, minimise the predictive disparity
  model$modelsense = "min"

  # the objective function is the prediction disparity, betas and u_j not in objective
  model$obj = c((1-S)/mean(1-S) - S/mean(S), rep(0, p+N))

  # set the linear constraint matrix
  model$A = rbind(cbind(diag(1, nrow = n), -XX, matrix(0, nrow = n, ncol = N)), # policies - betas <= 1
                  cbind(diag(1, nrow = n), -XX, matrix(0, nrow = n, ncol = N)), # policies - betas > 0
                  c(rep(1, n), rep(0, p+N)), # capacity constraint
                  c(rep(0, n+p), rep(1, N))) # constraint on the u_j (C) >= 1

  # set the rhs (with tolerance 1e-6)
  model$rhs = c(rep(1-tolerance, n), rep(tolerance, n), capacity_constraint, 1)

  # set the constraint directions
  model$sense = c(rep("<=", n), rep(">", n), "<=", ">=")

  # define the type of the variables
```

```r
  model$vtype = c(rep("B", n), rep("C", p), rep("B", N))

  # put bounds on the parameter space, for z_i and u_j  [0,1], for the betas [-1,1]
  model$ub =  c(rep(1, n+p+N))
  model$lb =  c(rep(0, n), rep(-1, p), rep(0, N))

  # set up the quadratic constraints (B)
  listnames = c(1:N)
  model$quadcon = sapply(listnames, function(x) NULL)

  # initiate a for loop to add the N=sqrt(n) constraints
  for (i in 1:N){

    # initiate the Qc matrix
    model$quadcon[[i]]$Qc = matrix(0, nrow = n+p+N, ncol = n+p+N) # rows and cols must = ncol of A
    # set coefficients in the column of the respective u_j
    model$quadcon[[i]]$Qc[which(S==0), n+p+i] = (alpha[i])*GS0[GS0!=0] # male welfare, omit female 0s in GS0
    model$quadcon[[i]]$Qc[which(S==1), n+p+i] = (1-alpha[i])*GS1[GS1!=0] # female welfare, omit male 0s in GS1


    # initiate q vector for the linear terms (optimal welfare W_bar)
    model$quadcon[[i]]$q = rep(0, n+p+N)
    # insert the respective welfare (- to bring it to lhs)
    model$quadcon[[i]]$q[n+p+i] = -W_bar[i]

    # set the rhs
    model$quadcon[[i]]$rhs = -tolerance

    # direction of constraint
    model$quadcon[[i]]$sense = ">="

  }

  if (is.na(start[1]) == F) model$start = start

  # set a list of parameters
  params = list(IntFeasTol = 1e-9, FeasibilityTol = 1e-9, TimeLimit = timelimit, # tolerance limits
                BarConvTol = exp(-2), # tolerance on the barrier between primal and dual solution
                Disconnected = 0, # degree of exploitation of (independent) submodels
                Heuristics = 0, # fraction of runtime spent on feasibility heuristics
                NodefileStart = 0.5) # max node memory before writing to hard drive

  # solve the model
  result = gurobi(model, params = params)

  # extract the values
  beta = result$x[(n+1):(n+p)] # the betas for deciding on the policy
  u = result$x[((n+p+1)):length(result$x)] # which u_j is equal to 1
  weight = 1-alpha[u == 1] # store the respective optimal weight alpha_j (1-alpha since alpha is the male weight)
  policies = apply(X, 1, function(x) ifelse(x%*%beta > 0, 1, 0)) # the estimated policies
  objval = result$objval # the minimised unfairness (prediction disparity)

  # return estimated betas, weight of the female group in the optimisation, policies and objective value
  return(list(beta = beta, weight_female = weight, policies = policies, objval = objval,
              result.x = result$x, time = result$runtime, gap = result$mipgap))

}
```

```r

################################################################################

# Estimation FPT Prediction Disparity

################################################################################

library(slam)
library(Matrix)
library(gurobi)
library(dplyr)

#### Preliminaries
load("./results/implementation.RData")
load("./results/pareto_frontier.RData")

Y = df_sample$Y
D = df_sample$D
S = df_sample$S

m1 = df_sample$m1
m0 = df_sample$m0
ps = df_sample$ps

X = df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D)
capacity_constraint = floor(0.33*nrow(X))

N = floor(sqrt(nrow(X)))
alpha = seq(from = 0.05, to = 0.95, length.out = N)



#### FPT Prediction Disparity Estimation

## Build start vector with the help of the pareto frontier values to help find an
    initial solution in the optimisation
X_start = as.matrix(cbind(1, S, X))

# "list comprehension" to extract the policies from the beta coefficients and the
    data
policies = t(apply(pareto_beta, 1, function(x) sapply(X_start%*%x, function(y)
    ifelse(y > 0, 1, 0))))

# compute the objective value, i.e., the unfairness generated by the respective
    policy
objective = apply(policies, 1, function(x) sum(x*(1-S)/mean(1-S)) - sum(x*S/mean(S)
    ))

# find the minimal objective; as there are multiple results, choose the one closest
     to equal weighting for female and male
minimal_obj = which(objective == min(objective))
minimal_obj = minimal_obj[which.min(abs(minimal_obj - N/2))]

# start value
start_pred_disp = c(policies[minimal_obj,], pareto_beta[minimal_obj,], ifelse(c(1:N
    ) == minimal_obj, 1, 0))

# optimise! after 10'000s gap of 3.72%
FPT_pred_disparity = optimisePredictionDisparity(Y, X = cbind(S, X), D, S, m0, m1,
    ps, alpha = alpha, tolerance = 1e-6,
                                                 W_bar = pareto_W_bar, capacity_
                                                     constraint, start = start_pred
                                                     _disp,
                                                 timelimit = 15000)


# save(list = c("FPT_pred_disparity"), file = "./results/fpt_disp.RData")
```

## B.5 FPT Prediction Disparity Absolute

```
##########################################################################

# Fair Policy Targeting (Prediction Disparity Absolute): Function

##########################################################################

# Y: outcome
# X: matrix of covariates
# D: treatment vector
# S: sensitive attribute vector
# ps: estimated propensity score
# m1: conditional mean of treated individuals
# m0: conditional mean of untreated individuals
# alpha: weight of the male welfare
# tolerance: slackness parameter (discreteness)
# W_bar: the pareto frontier
# capacity_constraint: maximal number of individuals to be treated
# start: possible starting value for the optimisation
# timelimit: maximum time spent on optimisation in seconds

optimisePredictionDisparityAbs = function(Y, X, D, S, m0, m1, ps, alpha = seq(from = 0.05, to = 0.95, length.out = N),
                                          tolerance = 1e-6, W_bar, capacity_constraint, start = NA, timelimit = 10000){

  # compute the doubly robust score (drs)
  G11_hat = (S/mean(S)) * ((D/ps)*(Y-m1)+m1) # treated females
  G01_hat = (S/mean(S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated females
  G10_hat = ((1-S)/mean(1-S)) * ((D/ps)*(Y-m1)+m1) # treated males
  G00_hat = ((1-S)/mean(1-S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated males

  GS1 = G11_hat - G01_hat # female drs
  GS0 = G10_hat - G00_hat # male drs

  # add intercept for beta0, and normalise
  X = as.matrix(cbind(1, X))
  max_val = max(apply(X, 1, function(x) sum(abs(x))))
  XX = X/max_val

  # set parameters
  n = nrow(XX)
  p = ncol(XX)

  ## initialise the model
  model = list()

  # sense of the optimisation, minimise the predictive disparity
  model$modelsense = "min"

  # the objective function is the abs prediction disparity, betas and u_j not in objective
  model$obj = c(rep(1, n), rep(0, p+N+2))

  # set the linear constraint matrix
  model$A = rbind(cbind(diag(1, nrow = n), -XX, matrix(0, nrow = n, ncol = N+2)), # policies - betas <= 1
                  cbind(diag(1, nrow = n), -XX, matrix(0, nrow = n, ncol = N+2)), # policies - betas > 0
                  c(rep(1, n), rep(0, p+N+2)), # capacity constraint
                  c(rep(0, n+p), rep(1, N), 0, 0), # constraint on the u_j (C) >= 1
                  c(((1-S)/mean(1-S) - S/mean(S)), rep(0, p+N), -1, 0), # policies - slack_var <= 0
                  c(-((1-S)/mean(1-S) - S/mean(S)), rep(0, p+N), 0, -1)) # -slack_var -policies <= 0

  # set the rhs (with tolerance 1e-6)
  model$rhs = c(rep(1-tolerance, n), rep(tolerance, n), capacity_constraint, 1, 0,
```

```
    0)

  # set the constraint directions
  model$sense = c(rep("<=", n), rep(">", n), "<=", ">=", "<=", "<=")

  # define the type of the variables
  model$vtype = c(rep("B", n), rep("C", p), rep("B", N), "C", "C")

  # put bounds on the parameter space, for z_i, constants, and u_j  [0,1], for the betas [-1,1], Inf for the slack variables
  model$ub =  c(rep(1, n+p+N), Inf, Inf)
  model$lb =  c(rep(0, n), rep(-1, p), rep(0, N+2))

  # set up the quadratic constraints (B)
  listnames = c(1:N)
  model$quadcon = sapply(listnames, function(x) NULL)

  # initiate a for loop to add the N=sqrt(n) constraints
  for (i in 1:N){

    # initiate the Qc matrix
    model$quadcon[[i]]$Qc = matrix(0, nrow = n+p+N+2, ncol = n+p+N+2) # rows and cols must = ncol of A
    # set coefficients in the column of the respective u_j
    model$quadcon[[i]]$Qc[which(S==0), n+p+i] = alpha[i]*GS0[GS0!=0] # male welfare, omit female 0s in GS0
    model$quadcon[[i]]$Qc[which(S==1), n+p+i] = (1-alpha[i])*GS1[GS1!=0] # female welfare, omit male 0s in GS1

    # initiate q vector for the linear terms (optimal welfare W_bar)
    model$quadcon[[i]]$q = rep(0, n+p+N+2)
    # insert the respective welfare (- as bring it to lhs)
    model$quadcon[[i]]$q[n+p+i] = -W_bar[i]

    # set the rhs
    model$quadcon[[i]]$rhs = -tolerance

    # direction of constraint
    model$quadcon[[i]]$sense = ">="

  }

  if (is.na(start[1]) == F) model$start = start

  # set a list of parameters
  params = list(IntFeasTol = 1e-9, FeasibilityTol = 1e-9, TimeLimit = timelimit, # tolerance limits
                BarConvTol = exp(-2), # tolerance on the barrier between primal and dual solution
                Disconnected = 0, # degree of exploitation of (independent) submodels
                Heuristics = 0, # fraction of runtime spent on feasibility heuristics
                NodefileStart = 0.5) # max node memory before writing to hard drive

  # solve the model
  result = gurobi(model, params = params)

  # extract the values
  beta = result$x[(n+1):(n+p)] # the betas for deciding on the policy
  u = result$x[((n+p+1)):length(result$x)] # which u_j is equal to 1
  weight = 1-alpha[u == 1] # store the respective optimal weight alpha_j
  policies = apply(X, 1, function(x) ifelse(x%*%beta > 0, 1, 0)) # the estimated policies
  objval = result$objval # the minimised unfairness (prediction disparity)

  # return estimated betas, weight of the female group in the optimisation, policies and objective value
  return(list(beta = beta, weight_female = weight, policies = policies, objval = objval,
```

```r
                result.x = result$x, time = result$runtime, gap = result$mipgap))

}

################################################################################

# Estimation FPT Prediction Disparity Absolute

################################################################################

library(slam)
library(Matrix)
library(gurobi)
library(dplyr)

#### Preliminaries
load("./results/implementation.RData")
load("./results/pareto_frontier.RData")

Y = df_sample$Y
D = df_sample$D
S = df_sample$S

m1 = df_sample$m1
m0 = df_sample$m0
ps = df_sample$ps

X = df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D)
capacity_constraint = floor(0.33*nrow(X))

N = floor(sqrt(nrow(X)))
alpha = seq(from = 0.05, to = 0.95, length.out = N)



#### FPT Prediction Disparity Absolute Estimation

## Build start vector with the help of the pareto frontier values to help find an initial solution in the optimisation
X_start = as.matrix(cbind(1, S, X))

# "list comprehension" to extract the policies from the beta coefficients and the data
policies = t(apply(pareto_beta, 1, function(x) sapply(X_start%*%x, function(y) ifelse(y > 0, 1, 0))))

# compute the objective value, i.e., the unfairness generated by the respective policy
objective = apply(policies, 1, function(x) abs( sum(x*(1-S)/mean(1-S)) - sum(x*S/mean(S)) ))

# find the minimal objective; as there are multiple results, choose the one closest to equal weighting for female and male
minimal_obj = which(objective == min(objective))
minimal_obj = minimal_obj[which.min(abs(minimal_obj - N/2))]

# start value
start_pred_disp_abs = c(policies[minimal_obj,], pareto_beta[minimal_obj,], ifelse(c(1:N) == minimal_obj, 1, 0), NA, NA)

# optimise! after 10'000s gap of 83.6%
FPT_pred_disparity_abs = optimisePredictionDisparityAbs(Y, X = cbind(S, X), D, S, m0, m1, ps, alpha = alpha, tolerance = 1e-6,
                                                        W_bar = pareto_W_bar, capacity_constraint, start = start_pred_disp_abs,
                                                        timelimit = 15000)


```

```
# save(list = c("FPT_pred_disparity_abs"), file = "./results/fpt_absolute_disp.
    RData")
```

## B.6 Regularised Welfare Maximisation

```
################################################################################

# Regularised Welfare Maximisation: Function

################################################################################

# Y: outcome
# X: matrix of covariates
# D: treatment vector
# S: sensitive attribute vector
# ps: estimated propensity score
# m1: conditional mean of treated individuals
# m0: conditional mean of untreated individuals
# alpha: weight of the male welfare
# tolerance: slackness parameter (discreteness)
# capacity_constraint: maximal number of individuals to be treated
# timelimit: maximum time spent on optimisation in seconds
# lambda: regularisation parameter

optimiseRegularised = function(Y, X, D, S, ps, m1, m0, alpha = mean(1-S), tolerance = 1e-6,
                               capacity_constraint, timelimit = 5000, lambda = 0.004){

  # load libraries
  library(Matrix)
  library(slam)
  library(gurobi)

  # compute the doubly robust score (drs)
  G11_hat = (S/mean(S)) * ((D/ps)*(Y-m1)+m1) # treated females
  G01_hat = (S/mean(S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated females
  G10_hat = ((1-S)/mean(1-S)) * ((D/ps)*(Y-m1)+m1) # treated males
  G00_hat = ((1-S)/mean(1-S)) * (((1-D)/(1-ps))*(Y-m0)+m0) # untreated males

  GS1 = G11_hat - G01_hat # female
  GS0 = G10_hat - G00_hat # male
  G = alpha*GS0 + (1-alpha)*GS1 # scaled vector

  # add intercept for beta0, and normalise
  X = as.matrix(cbind(1, X))
  max_val = max(apply(X, 1, function(x) max(abs(x))))
  XX = X/max_val

  # set parameters
  n = nrow(X)
  p = ncol(X)

  ## initialise the model
  model = list()

  # sense of the optimisation, maximise utility
  model$modelsense = "max"

  # objective function contains the welfare & regularisation components
  model$obj = c(G, rep(0, p), rep(-lambda, 2))

  # set the linear constraint matrix
  model$A = rbind(cbind(diag(1, nrow = n), -X, 0, 0), # policies - betas <= 1
                  cbind(diag(1, nrow = n), -X, 0, 0), # policies - betas > 0
                  c(rep(1, n), rep(0, p+2)), # treated <= capacity constraint
                  c(-((1-S)/mean(1-S) - S/mean(S)), rep(0, p), -1, 0), # -w - policies <= 0 (male - female)
                  c(((1-S)/mean(1-S) - S/mean(S)), rep(0, p), 0, -1)) # policies - w <= 0 (male - female)

  # the rhs of the constraints, with tolerance (1e-6)
  model$rhs = c(rep(1 - tolerance, n), rep(tolerance, n), capacity_constraint, rep
```

```
      (0, 2))

  # set the constraint directions
  model$sense = c(rep("<=", n), rep(">", n), rep("<=", 3))

  # define the type of the variables
  model$vtype = c(rep("B", n), rep("C", p+2))

  # Put bounds on the parameter space, for z [0,1] and for the betas [-1,1], Inf for the slack variables
  model$ub = c(rep(1, n+p), rep(Inf, 2))
  model$lb = c(rep(0, n), rep(-1, p), rep(0, 2))

  # set additional parameters for the optimisation
  params = list(IntFeasTol = 1e-9, FeasibilityTol = 1e-9, TimeLimit = timelimit, # tolerance limits
                BarConvTol = exp(-2), # tolerance on the barrier between primal and dual solution
                Disconnected = 0, # degree of exploitation of (independent) submodels
                Heuristics = 0, # fraction of runtime spent on feasibility heuristics
                NodefileStart = 0.5) # max node memory before writing to hard drive

  # solve the model
  result = gurobi(model, params = params)

  # extract the values
  beta = result$x[(n+1):(n+p)]
  policies = apply(X, 1, function(x) ifelse(x%*%beta > 0, 1, 0))
  objval = result$objval

  return(list(beta = beta, policies = policies, objval = objval, lambda = lambda,
              result.x = result$x, time = result$runtime, gap = result$mipgap))

}



###############################################################################

# Estimation FPT Regularised

###############################################################################

library(slam)
library(Matrix)
library(gurobi)
library(dplyr)

#### Preliminaries
load("./results/implementation.RData")

Y = df_sample$Y
D = df_sample$D
S = df_sample$S

m1 = df_sample$m1
m0 = df_sample$m0
ps = df_sample$ps

X = df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D)
capacity_constraint = floor(0.33*nrow(X))



#### Regularised Estimation

# regularisation with lambda = 0.004
FPT_regularised_004 = optimiseRegularised(Y, X = cbind(S,X), D, S, ps, m1, m0,
```

```
    alpha = mean(1-S), tolerance = 1e-6,
                                          capacity_constraint, timelimit = 15000,
                                              lambda = 0.004)

# regularisation with lambda = 0.010
FPT_regularised_010 = optimiseRegularised(Y, X = cbind(S,X), D, S, ps, m1, m0,
    alpha = mean(1-S), tolerance = 1e-6,
                                          capacity_constraint, timelimit = 15000,
                                              lambda = 0.010)

# regularisation with lambda = 0.100
FPT_regularised_100 = optimiseRegularised(Y, X = cbind(S,X), D, S, ps, m1, m0,
    alpha = mean(1-S), tolerance = 1e-6,
                                          capacity_constraint, timelimit = 15000,
                                              lambda = 0.100)


# save(list = c("FPT_regularised_004", "FPT_regularised_010", "FPT_regularised
    _100"), file = "./results/fpt_regularised.RData")

# share of treated male / female students with the different approaches
gf = as_tibble(cbind(FPT_regularised_004$policies, FPT_regularised_010$policies,
    FPT_regularised_100$policies, S))
treatment_dist = gf %>% group_by(S) %>% summarise(D004 = mean(V1), D010 = mean(V2),
     D100 = mean(V3))
```

## B.7 Frontier Plot

```
##################################################################################

# Estimation of the Pareto Frontier for the Frontier Plots

##################################################################################

#### Prepare data and sample
library(dplyr)
load("./results/implementation.RData")

Y = df_sample$Y
D = df_sample$D
S = df_sample$S

m1 = df_sample$m1
m0 = df_sample$m0
ps = df_sample$ps

X = df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D)
capacity_constraint = floor(0.33*nrow(X))

N = 100
alpha = seq(from = 0.05, to = 0.95, length.out = N)


# load libraries
library(slam)
library(gurobi)
library(foreach)
library(future)
library(doFuture)
plan(multisession)

# policy 1 and female >= male
frontier_P1_fem = foreach(i = alpha, .combine = rbind) %dofuture% {

  result = EWM_estimation(Y, X = cbind(S,X), D, S, ps, m1, m0, alpha = i, tolerance
      = 1e-3,
                          capacity_constraint, funclass3 = T, parity_sense = ">=",
                              timelimit = 500)

  c(result[[2]], result[[3]], result[[1]])
}

# policy 1 and female <= male
frontier_P1_male = foreach(i = alpha, .combine = rbind) %dofuture% {

  result = EWM_estimation(Y, X = cbind(S,X), D, S, ps, m1, m0, alpha = i, tolerance
      = 1e-3,
                          capacity_constraint, funclass3 = T, parity_sense = "<=",
                              timelimit = 500)

  c(result[[2]], result[[3]], result[[1]])
}

# policy 2 (not using S in decision rule) and female >= male
frontier_P2_fem = foreach(i = alpha, .combine = rbind) %dofuture% {

  result = EWM_estimation(Y, X, D, S, ps, m1, m0, alpha = i, tolerance = 1e-3,
                          capacity_constraint, funclass3 = T, parity_sense = ">=",
                              timelimit = 500)

  c(result[[2]], result[[3]], result[[1]])
}

# policy 2 (not using S in decision rule) and female <= male
frontier_P2_male = foreach(i = alpha, .combine = rbind) %dofuture% {

```

```
  result = EWM_estimation(Y, X, D, S, ps, m1, m0, alpha = i, tolerance = 1e-3,
                          capacity_constraint, funclass3 = T, parity_sense = "<=",
                              timelimit = 500)

  c(result[[2]], result[[3]], result[[1]])
}

# save(list = c("frontier_P1_fem", "frontier_P1_male", "frontier_P2_fem", "frontier
    _P2_male"), file = "./results/frontier_plot.RData")

################################################################################

# Pareto Frontier Plots

################################################################################

# load library for plot
library(ggplot2)
library(ggthemes)
library(extrafont)
font_import()

# function for the comparison of the welfares and detection of pareto dominant
    allocations;
# corresponding largely to Viviano & Bradic's (2024) function in supplementary
    materials
pareto_dominance = function(female_function, male_function){

  # combine the solutions into matrix
  combined = rbind(female_function, male_function)

  # dummy vector indicating whether an allocation is not dominated
  dominated = rep(0, nrow(combined))

  # initiate for loops, compare each allocation i against all others j
  for(i in 1:nrow(combined)){
    indicator = 0
    for(j in 1:nrow(combined)){
      if(i != j){
        # dominated is 1 if i < j for both, female and male individuals, otherwise
            0
        indicator = max(combined[i,1] < combined[j,1] & combined[i,2] < combined[j
            ,2], indicator)
      }
    }
    # vector position of dominated allocations
    dominated[i] = indicator
  }

  # return only non dominated allocations
  return(combined[dominated == 0,])
}


# difference of the doubly robust scores for female and male (welfare improvement)
# (multiply with policy if pi = 1; otherwise only baseline welfare)
Wfem = S/mean(S) * (((D/ps)*(Y-m1)+m1) - ((1-D)/(1-ps)*(Y-m0)+m0))
Wmale = (1-S)/mean(1-S) * (((D/ps)*(Y-m1)+m1) - ((1-D)/(1-ps)*(Y-m0)+m0))

# baseline effect, i.e. doubly robust score of untreated individuals
# (multiply with S=s/mean(S=s) )
baseline_fem = S/mean(S) * ((1-D)/(1-ps)*(Y-m0)+m0)
baseline_male = (1-S)/mean(1-S) * ((1-D)/(1-ps)*(Y-m0)+m0)

# covariates as matrix; dimensions of the matrix
X = as.matrix(df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D))
n = nrow(X)
p = ncol(X)+1

# data frame with the covariates for female / male
```

```
df_fem = as.matrix(cbind(1,1,X))
df_male = as.matrix(cbind(1,0,X))

# load the results of the frontier computations, store the betas
load("./results/frontier_plot.RData")
beta_P1_fem = frontier_P1_fem[, (n+1):(n+p+1)] # p+1 due to added S
beta_P1_male = frontier_P1_male[, (n+1):(n+p+1)] # p+1 due to added S
beta_P2_fem = frontier_P2_fem[, (n+1):(n+p)]
beta_P2_male = frontier_P2_male[, (n+1):(n+p)]


# Compute the welfare

# Policy function class 1

# with female >= male
welfare_fem1 = apply(beta_P1_fem, 1, function(x) mean(sapply(df_fem%*%x, function(y
    ) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem))
welfare_male1 = apply(beta_P1_fem, 1, function(x) mean(sapply(df_male%*%x, function
    (y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male))
welfare_fem_function = cbind(welfare_fem1, welfare_male1)

# with female <= male
welfare_fem2 = apply(beta_P1_male, 1, function(x) mean(sapply(df_fem%*%x, function(
    y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem))
welfare_male2 = apply(beta_P1_male, 1, function(x) mean(sapply(df_male%*%x,
    function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male))
welfare_male_function = cbind(welfare_fem2, welfare_male2)

# compute pareto frontier
welfare1 = pareto_dominance(welfare_fem_function, welfare_male_function)
# add minimum values for nicer plot
welfare1 = rbind(c(0.4, max(welfare1[,2])), welfare1, c(max(welfare1[,1]), 0.45))


# Policy function class 2

# with female >= male
welfare_fem1 = apply(beta_P2_fem, 1, function(x) mean(sapply(df_fem[,-2]%*%x,
    function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem))
welfare_male1 = apply(beta_P2_fem, 1, function(x) mean(sapply(df_male[,-2]%*%x,
    function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male))
welfare_fem_function = cbind(welfare_fem1, welfare_male1)

# with female <= male
welfare_fem2 = apply(beta_P2_male, 1, function(x) mean(sapply(df_fem[,-2]%*%x,
    function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem))
welfare_male2 = apply(beta_P2_male, 1, function(x) mean(sapply(df_male[,-2]%*%x,
    function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male))
welfare_male_function = cbind(welfare_fem2, welfare_male2)

# compute pareto frontier
welfare2 = pareto_dominance(welfare_fem_function, welfare_male_function)
# add minimum values for nicer plot
welfare2 = rbind(c(0.4, max(welfare2[,2])), welfare2, c(max(welfare2[,1]), 0.45))


# Policy function class 3

# only female >= male
welfare3 = cbind(welfare_fem1, welfare_male1)
# add minimum values for nicer plot
welfare3 = rbind(c(0.4, max(welfare3[,2])), welfare3, c(max(welfare3[,1]), 0.45))


# name the allocations after the policy function class
policy_class = c(rep("Class 1", dim(welfare1)[1]), rep("Class 2", dim(welfare2)[1])
    , rep("Class 3", dim(welfare3)[1]))

# create tibble
```

```
df_pareto = as.data.frame(cbind(rbind(welfare1, welfare2, welfare3), policy_class))

# update column names
names(df_pareto) = c('Wfemale', 'Wmale', 'Class')

# coerce columns from character to numeric
df_pareto = transform(df_pareto, Wfemale = as.numeric(Wfemale), Wmale = as.numeric(
    Wmale))


# create plot
ggplot(data = df_pareto, aes(x = Wfemale, y = Wmale)) +
  geom_line(aes(color = Class), linetype = 1, linewidth = 0.7, show.legend = F) +
  geom_ribbon(aes(ymin = min(Wmale), ymax = Wmale, fill = Class)) +
  scale_color_manual(values=c("#69b3a2", "#404080", "darkgrey")) +
  scale_fill_manual(values = alpha(c("#69b3a2", "#404080", "darkgrey"), 0.4),
                    labels = c("Policy Class 1", "Policy Class 2", "Policy Class 3"
                        ),
                    name = "Policy Function Class") +
  xlab("Welfare female individuals") +
  ylab("Welfare male individuals") +
  theme_hc() +
  theme(text = element_text(family = "Times New Roman"),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 12),
        legend.title = element_text(size = 16),
        legend.text = element_text(size = 12))

ggsave("frontier_plot.png", plot = last_plot(), path = "./results/", height = 7,
    width = 12)
```

## B.8 Welfare Comparison Table

```
##############################################################################

# Welfare Comparison: Empirical Application

##############################################################################

load("./results/implementation.RData")

Y = df_sample$Y
D = df_sample$D
S = df_sample$S

m1 = df_sample$m1
m0 = df_sample$m0
ps = df_sample$ps

# difference of the doubly robust scores for female and male (welfare improvement)
# (multiply with policy if pi = 1; otherwise only baseline welfare)
Wfem = S/mean(S) * (((D/ps)*(Y-m1)+m1) - ((1-D)/(1-ps)*(Y-m0)+m0))
Wmale = (1-S)/mean(1-S) * (((D/ps)*(Y-m1)+m1) - ((1-D)/(1-ps)*(Y-m0)+m0))

# baseline effect, i.e. doubly robust score of untreated individuals
# (multiply with S=s/mean(S=s) )
baseline_fem = S/mean(S) * ((1-D)/(1-ps)*(Y-m0)+m0)
baseline_male = (1-S)/mean(1-S) * ((1-D)/(1-ps)*(Y-m0)+m0)

# covariates as matrix
X = as.matrix(df_sample %>% select(C3, X1_D, X2_D, X3_D, X4_D, X5_D))

# data frame with the covariates for female / male
df_fem = as.matrix(cbind(1,1,X))
df_male = as.matrix(cbind(1,0,X))

##############################################################################
# Envy
load("./results/fpt_counterfactual_envy.RData")

# betas for policies and weight of female group
beta_envy = FPT_counterfactual_envy$beta
weight_envy = FPT_counterfactual_envy$weight_female
# welfare under the envy measure for female and male individuals
# note: matrix multiplication to extract policy, multiply with welfare components
welfare_fem_envy = mean(sapply(df_fem%*%beta_envy, function(y) ifelse(y >= 0, 1, 0)
    )*Wfem + baseline_fem)
welfare_male_envy = mean(sapply(df_male%*%beta_envy, function(y) ifelse(y >= 0, 1,
    0))*Wmale + baseline_male)


##############################################################################
# Prediction Disparity
load("./results/fpt_disp.RData")

# betas for policies and weight of female group
beta_pred_disp = FPT_pred_disparity$beta
weight_pred_disp = FPT_pred_disparity$weight_female
# welfare under the envy measure for female and male individuals
# note: matrix multiplication to extract policy, multiply with welfare components
welfare_fem_pred_disp = mean(sapply(df_fem%*%beta_pred_disp, function(y) ifelse(y >
    = 0, 1, 0))*Wfem + baseline_fem)
welfare_male_pred_disp = mean(sapply(df_male%*%beta_pred_disp, function(y) ifelse(y
     >= 0, 1, 0))*Wmale + baseline_male)


##############################################################################
# Prediction Disparity Absolute
load("./results/fpt_absolute_disp.RData")

# betas for policies and weight of female group
```

```
beta_pred_disp_abs = FPT_pred_disparity_abs$beta
weight_pred_disp_abs = FPT_pred_disparity_abs$weight_female
# welfare under the envy measure for female and male individuals
# note: matrix multiplication to extract policy, multiply with welfare components
welfare_fem_pred_disp_abs = mean(sapply(df_fem%*%beta_pred_disp_abs, function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem)
welfare_male_pred_disp_abs = mean(sapply(df_male%*%beta_pred_disp_abs, function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male)


##############################################################################
# Regularisation Approach
load("./results/fpt_regularised.RData")

# betas for policies
beta_004 = FPT_regularised_004$beta
beta_010 = FPT_regularised_010$beta
beta_100 = FPT_regularised_100$beta

# welfare under the envy measure for female and male individuals
# note: matrix multiplication to extract policy, multiply with welfare components
welfare_fem_regularised_004 = mean(sapply(df_fem%*%beta_004, function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem)
welfare_male_regularised_004 = mean(sapply(df_male%*%beta_004, function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male)

welfare_fem_regularised_010 = mean(sapply(df_fem%*%beta_010, function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem)
welfare_male_regularised_010 = mean(sapply(df_male%*%beta_010, function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male)

welfare_fem_regularised_100 = mean(sapply(df_fem%*%beta_100, function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem)
welfare_male_regularised_100 = mean(sapply(df_male%*%beta_100, function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male)


##############################################################################
# EWM
load("./results/EWM_estimation.RData")

# betas for policies
beta_P1 = EWM_P1$beta
beta_P2 = EWM_P2$beta
beta_P3 = EWM_P3$beta

# welfare under the envy measure for female and male individuals
# note: matrix multiplication to extract policy, multiply with welfare components
welfare_fem_EWM_P1 = mean(sapply(df_fem%*%beta_P1, function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem)
welfare_male_EWM_P1 = mean(sapply(df_male%*%beta_P1, function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male)

welfare_fem_EWM_P2 = mean(sapply(cbind(1,X)%*%beta_P2, function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem)
welfare_male_EWM_P2 = mean(sapply(cbind(1,X)%*%beta_P2, function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male)

welfare_fem_EWM_P3 = mean(sapply(cbind(1,X)%*%beta_P3, function(y) ifelse(y >= 0, 1, 0))*Wfem + baseline_fem)
welfare_male_EWM_P3 = mean(sapply(cbind(1,X)%*%beta_P3, function(y) ifelse(y >= 0, 1, 0))*Wmale + baseline_male)


##############################################################################

## Combine all elements

# combine the welfares in a vector
welfares_fem = round(c(welfare_fem_envy, welfare_fem_pred_disp, welfare_fem_pred_
```

```
    disp_abs,
                 welfare_fem_EWM_P1, welfare_fem_EWM_P2, welfare_fem_EWM_P3,
                 welfare_fem_regularised_004, welfare_fem_regularised_010, welfare_
                     fem_regularised_100),3)

welfares_male = round(c(welfare_male_envy, welfare_male_pred_disp, welfare_male_
    pred_disp_abs,
                  welfare_male_EWM_P1, welfare_male_EWM_P2, welfare_male_EWM_P3,
                  welfare_male_regularised_004, welfare_male_regularised_010,
                      welfare_male_regularised_100),3)


# the importance weights (weight for EWM is mean(S) by definition); lambda for
    regularised approach
weights = round(c(weight_envy, weight_pred_disp, weight_pred_disp_abs,
                  rep(mean(S), 3), 0.004, 0.010, 0.100),3)

times = round(c(FPT_counterfactual_envy$time, FPT_pred_disparity$time, FPT_pred_
    disparity_abs$time,
                EWM_P1$time, EWM_P2$time, EWM_P3$time,
                FPT_regularised_004$time, FPT_regularised_010$time, FPT_regularised
                    _100$time),1)

gaps = round(c(FPT_counterfactual_envy$gap, FPT_pred_disparity$gap, FPT_pred_
    disparity_abs$gap,
               EWM_P1$gap, EWM_P2$gap, EWM_P3$gap,
               FPT_regularised_004$gap, FPT_regularised_010$gap, FPT_regularised
                   _100$gap)*100, 3)

## Construct table
final_table = cbind(welfares_fem, welfares_male, weights, times, gaps)

colnames(final_table) = c("Welfare Female", "Welfare Male", "Importance Weight", "
    Time (s)", "Gap (%)")

rownames(final_table) = c("Counterfactual Envy", "Prediction Disparity", "
    Prediction Disparity Abs",
                          paste("Welfare Max.", c(1:3)), paste("Regularised", c
                              (1:3)))

# xtable(final_table)
```

## B.9 Unfairness Comparison Plot

```
################################################################################

# Unfairness Plots

################################################################################

# load required libraries
library(tidyverse)
library(ggthemes)
library(extrafont)
font_import()

# load the required results
load("./results/implementation.RData")
load("./results/fpt_counterfactual_envy.RData")
load("./results/fpt_disp.RData")
load("./results/fpt_absolute_disp.RData")
load("./results/fpt_regularised.RData")
load("./results/EWM_estimation.RData")

# some preliminary parametere definitions
Y = df_sample$Y
D = df_sample$D
S = df_sample$S

# Unfairness comparison with V = prediction disparity

# V with the prediciton disparity objective
V_pred_disp = (sum((1-S)*FPT_pred_disparity$policies)/sum(1-S)
               - sum(S*FPT_pred_disparity$policies)/sum(S))

# V with the EWM method
V_EWM1 = (sum((1-S)*EWM_P1$policies)/sum(1-S)
          - sum(S*EWM_P1$policies)/sum(S))

V_EWM2 = (sum((1-S)*EWM_P2$policies)/sum(1-S)
          - sum(S*EWM_P2$policies)/sum(S))

V_EWM3 = (sum((1-S)*EWM_P3$policies)/sum(1-S)
          - sum(S*EWM_P3$policies)/sum(S))

# V with the regularised approach
V_regularised004 = (sum((1-S)*FPT_regularised_004$policies)/sum(1-S)
                    - sum(S*FPT_regularised_004$policies)/sum(S))

V_regularised010 = (sum((1-S)*FPT_regularised_010$policies)/sum(1-S)
                    - sum(S*FPT_regularised_010$policies)/sum(S))

V_regularised100 = (sum((1-S)*FPT_regularised_100$policies)/sum(1-S)
                    - sum(S*FPT_regularised_100$policies)/sum(S))


# Unfairness with V = prediction disparity absolute

# Prediction disparity absolute comparison
V_pred_disp_abs = abs( sum((1-S)*FPT_pred_disparity_abs$policies)/sum(1-S)
                       - sum(S*FPT_pred_disparity_abs$policies)/sum(S) )

# V with the EWM method
V_EWM1_abs = abs( sum((1-S)*EWM_P1$policies)/sum(1-S)
                  - sum(S*EWM_P1$policies)/sum(S) )

V_EWM2_abs = abs( sum((1-S)*EWM_P2$policies)/sum(1-S)
                  - sum(S*EWM_P2$policies)/sum(S) )

V_EWM3_abs = abs( sum((1-S)*EWM_P3$policies)/sum(1-S)
                  - sum(S*EWM_P3$policies)/sum(S) )

```

```
# V with the regularised approach
V_regularised004_abs = abs( sum((1-S)*FPT_regularised_004$policies)/sum(1-S)
                           - sum(S*FPT_regularised_004$policies)/sum(S) )

V_regularised010_abs = abs( sum((1-S)*FPT_regularised_010$policies)/sum(1-S)
                           - sum(S*FPT_regularised_010$policies)/sum(S) )

V_regularised100_abs = abs( sum((1-S)*FPT_regularised_100$policies)/sum(1-S)
                           - sum(S*FPT_regularised_100$policies)/sum(S) )


# combine into dataframe
V = c(V_pred_disp, V_EWM1, V_EWM2, V_EWM3, V_regularised004, V_regularised010,
      V_regularised100, V_pred_disp_abs, V_EWM1_abs, V_EWM2_abs, V_EWM3_abs,
      V_regularised004_abs, V_regularised010_abs, V_regularised100_abs)

names(V) = rep(c("Prediction Disparity", paste("Welfare Max.", c(1:3)),
             paste("Regularised", c(1:3))),2)

df_V = enframe(V)

df_V$metric = as.factor(c(rep("Prediction Disparity", 7), rep("Prediction Disparity
     Absolute", 7)))

df_V = df_V %>%
  mutate(name = factor(name, levels = c("Prediction Disparity", paste("Welfare Max.
      ", c(1:3)),
                                        paste("Regularised", c(1:3)))))

# create the plot
ggplot(df_V, aes(x = metric, y = value, fill = name, colour = name)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.6) +
  scale_y_continuous(breaks = c(0.15, 0, -0.15, -0.30, -0.45, -0.6))+
  scale_fill_manual(values=c("#507872", "#69b3a2", "#acc6aa", "darkgrey",
                             "#595959", "#6a5acd", "#404080"), name = "Method")+
  scale_colour_manual(values=alpha(c("#507872", "#69b3a2", "#acc6aa", "darkgrey",
                               "#595959", "#6a5acd", "#404080"), 0.6), guide = "
                                   none")+
  xlab("") +
  ylab("Unfairness Level")+
  theme_hc()+
  theme(text = element_text(family = "Times New Roman"),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 12),
        legend.title = element_text(size = 16),
        legend.text = element_text(size = 12))


ggsave("unfairness.png", plot = last_plot(), path = "./results/", height = 7, width
     = 12)
```

# Declaration of Aids

| Aid | Usage | Affected parts |
|---|---|---|
| DeepL | Translation and grammar check | Occasional text passages |
| Grammarly web extension | Spellcheck | Whole paper |
| ChatGPT | Improve latex code of tables and equations | Tables and equations |
| Various R packages xtable, ggplot2, ggthemes, extrafont | Latex code for tables (xtable) and beautify graphs | Tables and graphs |

# Declaration of Authorship

I hereby declare,

- that I have written this thesis independently;
- that I have written the thesis using only the aids specified in the index;
- that all parts of the thesis produced with the help of aids have been precisely declared;
- that I have mentioned all sources used and cited them correctly according to established academic citation rules;
- that I have acquired all immaterial rights to any materials I may have used, such as images or graphics, or that these materials were created by me;
- that the topic, the thesis or parts of it have not already been the object of any work or examination of another course, unless this has been expressly agreed with the faculty member in advance and is stated as such in the thesis;
- that I am aware of the legal provisions regarding the publication and dissemination of parts or the entire thesis and that I comply with them accordingly;
- that I am aware that my thesis can be electronically checked for plagiarism and for third-party authorship of human or technical origin and that I hereby grant the University of St.Gallen the copyright according to the Examination Regulations as far as it is necessary for the administrative actions;
- that I am aware that the University will prosecute a violation of this Declaration of Authorship and that disciplinary as well as criminal consequences may result, which may lead to expulsion from the University or to the withdrawal of my title.

By submitting this thesis, I confirm through my conclusive action that I am submitting the Declaration of Authorship, that I have read and understood it, and that it is true.

....................................
St.Gallen, May 21, 2024