Propensity Score Matching for Multiple Treatments in Causal Inference*

Yiliu Cao

Abstract

Propensity scores (PS), which measures the probability of receiving a treatment given a set of covariates, are widely used to estimate average treatment effects with one of the most popular method is propensity score matching. In traditional binary treatments setting, units from treatment and control groups are matched based on similar propensity scores. However, in practice, it is common to have multiple treatments, making the propensity score matching and estimation of average treatment effects be more challenging as propensity scores is now a vector instead of a scalar. This paper will mainly focus on the matching methods on propensity scores under multi-treatment such as Vector Matching and its extensions. This paper will conduct a simulation study to evaluate the performance of various matching strategies in terms of covariate balance, and assess the accuracy of ATE estimation under these approaches.

1 Introduction

Causal inference plays an important role in many scientific fields, including epidemiology, finance, and more. It enables us to estimate the effect of specific interventions or treatments on outcomes we are interested in. Traditionally, causal inference focuses on binary treatments and compares the difference in outcomes between the treatment and control groups. However, in real-world applications, it is common that the outcome may not depend on only one treatment but on two or more treatments such as multi-drug testing. If we only investigate the single-treatment effect of one drug, then the study will be less comprehensive and convinced. In this paper, we are going to investigate the average treatment effect ATE under multi-treatment setting.

The main method that will be discussed in this paper is called the propensity score matching PSM. Propensity score is the probability of receiving the treatment given a set of covariates. In PSM, we firstly calcualte the propensity scores for each unit and then match the units from different treatment groups and then estimate the ATE by the difference of the outcome. However, unlike the traditional binary treatment where the propensity score is a scalar, the propensity becomes a vector (so-called the generalized propensity scores GPS) under the multi-treatment. This makes the matching more difficult as we GPS for each unit is a vector instead. It would be nearly impossible to find two units from different treatment but has the same GPS vector. In this paper, we will discuss the method called Vector Matching VM which is PSM for GPS. We will especially focus on the balance of covariates before and after matching, and most importantly, we will see how to estimate the ATE between different pair of treatment and see the accuracy of estimation.

There are plenty of researches focusing on the PSM for GPS. In terms of methodologies estimating the GPS, the most common way is to use the multinomial logistic regressions (McCaffrey et al.,

*All codes and data analyses can be found at: https://github.com/yiliuc/psm_multi-treatments

2013). In addition to the parametric regression, two machine-learning based methods, generalized boosted model (GBM) and Random Forests (RF), can be also applied to estimate the GPS. For GBM, it consists many simple regression trees iteratively and implement separate GBMs on each indicator of each treatment to estimate the propensity scores, respectively (McCaffrey et al., 2013). Zhu et al. (Zhu et al., 2015) further extends this idea to multiple continuous treatments. In terms of GBM, RF is a collection of classification and regression trees CART, and the GPS is predicted as the fraction of trees that classify the unit to the certain treatment (Lee et al., 2010; Lee et al., 2011). In terms of the matching method for GPS, it is quite a recent topic. The idea is insighted by the method called Series of binomial comparisions by Lechner (Lechner, 2001), and was formalized to PSM for GPS, called Vector Matching by Lopez and Gutman (Lopez & Gutman, 2017). The key idea for VM is that we use clustering methods such as K-means clustering on the rest propensity scores while perform matching. VM was further extended by including, for example Mahalanobis distance or fuzzy clustering, by Scotina and Gutman (Scotina & Gutman, 2019). This topic has rich potential to be extended further, and this paper will discuss the original algorithm and its two extensions to test their performance on estimating the ATE.

This paper is structured as follows. Section 2 will take a brief review of causal inference and generalized propensity scores. We will presents the detailed Vector Matching Algorithm and two extensions on Section 3. In Section 4, we will conduct a simulation study to demonstrate its performance by 1) checking the balance of matching and 2) the accuracy of estimation of ATEs under the matching methods. All the results, as well as the limitation and future work will be discussed in Section 5.

2 Framework

2.1 Notations for Multi-treatment

What distinguished the causal inference from the traditional inference is that it focuses on uncovering causal relationships between treatments and outcomes, rather than merely identifying associations. While association examines correlations or patterns between covariates and outcomes, causal inference seeks to determine whether changes in a treatment variable actually cause changes in the outcome. However, the key problem in estimating the causal effects is the fundamental problem which states that we can only observe the outcome under the treatment they actually received, not under alternative treatments. For instance, if a patient receives medication A, we cannot observe what the outcome would have been had they instead received medication B.

To address the fundamental problem, causal inference relies on two key assumptions. The first one is the consistency assumption, which states that if a subject receive the treatment t, then his observed outcome Y is equal to the potential outcome under that treatment, denoted Y^t. Formally, consistecny assumes $Y = Y^t$. The second assumption is the exchangebility, which assumes the potential outcomes are independent of the treatment assignment. That is, $T \perp Y^t$. These two assumptions are foundational and apply to both binary and multi-treatment settings in causal inference framework.

Under these two assumptions, let $Y_i, \mathbf{X}_i, T_i$ denote the observed outcome, set of covariates and treatment assignment for each subject $i = 1, ..., N$ with $N < \mathcal{N}$ where $\mathcal{N}$ is the population size. Let $\mathcal{T}$ be the treatment space of Z treatments such that $|\mathcal{T}| = Z$. In addition, let Y_i^j be the potential outcome of receiving treatment j for subject i, and $\mathbf{Y}_i = \{Y_i^j\}_{j=1}^Z$ represent the full vector of potential outcomes for subject i under each treatment.

In terms of average treatment effect ATE and the average treatment effect on the treated ATT in the context of multiple treatments, let $\mathcal{T} = \{T_j\}_{j=1}^{Z}$, let $w_1, w_2 \subseteq \mathcal{T}$ be two subgroups of treatments such that $w_1 \cap w_2 = \varnothing$. Then ACE between these two groups of treatment $ACT_{w_1,w_2}$is defined as

$$ATE_{w_1,w_2} = E\left[\frac{\sum_{t\in w_1} Y_i^t}{|w_1|} - \frac{\sum_{t\in w_2} Y_i^t}{|w_2|}\right]$$

and the average treatment effect on the treated among units receiving treatments in w_1 is defined as

$$ATT_{w_1|w_1,w_2} = E\left[\frac{\sum_{t\in w_1} Y_i^t}{|w_1|} - \frac{\sum_{t\in w_2} Y_i^t}{|w_2|}\middle|\, T_i \in w_1\right]$$

In this paper, we will focus only on the case where where $|w_1| = |w_2| = 1$, i.e., comparisons between two specific treatments. For example, if $\mathcal{T} = \{1, 2, 3\}$ such that $|\mathcal{T}| = 3$, the average treatment effect between treatment 1 and 2 and the corresponding average treatment effect among units receiving 1 are defined as

$$ATE_{12} = E\left[Y_i^1 - Y_i^2\right] \quad ATT_{1|12} = E\left[Y_i^1 - Y_i^2 |\, T_i = 1\right]$$

2.2 Propensity Scores

As previously introduced, the propensity score refers to the probability of receiving a specific treatment conditional on a set of covariates $\mathbf{X}$. For a single treatment t, he propensity score is defined as

$$r(t, \mathbf{X}) = p(T = t \mid \mathbf{X}) \in \mathbb{R}$$

In the case of multiple treatments, each unit has a nonzero probability of receiving any one of the Z treatments. Consequently, the propensity score generalize to a vector form, which is so-called as generalized propensity scores GPS. The GPS is defined as

$$\mathbf{R}(\mathbf{X}) = \left(r\left(t_1, \mathbf{X}\right), \ldots, r\left(t_Z, \mathbf{X}\right)\right) \in \mathbb{R}^Z$$

Propensity scores are widely used in causal inference due to two key properties that make it a powerful tool for addressing confounding. First, the propensity score is a balancing score, which implies that conditional on the propensity score, the distribution of observed covariates is independent of the treatment assignment. This property ensures that units with the same propensity score have similar distributions of covariates, regardless of the treatment received. As a result, propensity scores effectively reduce confounding bias by adjusting for covariates that influence both treatment assignment and outcomes. In addition, if treatment assignment is independent of the potential outcomes conditional on covariates, then this independence also holds conditional on the propensity score. This implies the causal estimand $E[Y^a|r(t, \mathbf{X})]$ identifiable, and allow us to estimate ATE or ATT. The proof of the two properties can be found in Appendix 6.1.

As introduce previously, the methods of estimating propensity scores are various include parametric regression such as multinomial logistic regression (McCaffrey et al., 2013), and machine-learning based methods such as generalized boosted models GBM and random forest RF (Lee et al., 2010; Lee et al., 2011; McCaffrey et al., 2013). In addition, the choice of estimation method also depends on the nature of the treatment. In general, there are two types of treatment which are ordinal and nominal treatments. The ordinal treatments describes the treatments that follows certain order such as low, medium and high, while there are not any orders for nominal treatments (Lopez & Gutman, 2017). In this paper, we will only consider the nominal treatments. Since the primary focus on

this paper is not the estimation of propensity scores, we will use multinomial logistic regression for simplicity and interpretability.

Multinomial logistic regression [McCaffrey et al. (2013);]estimates each propensity score by modeling the log-odds of each treatment relative to a reference category, typically the last treatment Z. For $t = 1, \ldots, Z-1$, it assumes

$$\log\left(\frac{P\left(T=t \mid \mathbf{X}\right)}{P\left(T=Z \mid \mathbf{X}\right)}\right) = \mathbf{X}'\boldsymbol{\beta}_z$$

Then corresponding propensity score for each treatment is estimated as

$$P\left(T=t \mid \mathbf{X}\right) = \frac{\exp\left(\mathbf{X}'\boldsymbol{\beta}_t\right)}{1+\sum_{j=1}^{Z-1}\exp\left(\mathbf{X}'\boldsymbol{\beta}_j\right)}, \quad t = 1, \ldots, Z-1$$

$$P\left(T=Z \mid \mathbf{X}\right) = \frac{1}{1+\sum_{j=1}^{Z-1}\exp\left(\mathbf{X}'\boldsymbol{\beta}_j\right)}$$

The R codes on estimating the generalized propensity scores can be found at Appendix 6.5.2.

3 Matching on Generalized Propensity Scores

The primary propensity score-based method considered in this paper is propensity score matching. It says that, if we can match the units from different treatment groups, who has the similar propensity scores, then we can use these matched pair of units to estimate the treatment effect by the average difference on the outcome. This method is straightforward to implement in binary treatment, where the propensity score for each unit is just a scalar. However, as the number of treatments increases, the propensity scores now become a vector. If we only match the unit based on the propensity score for one or several treatments, it does not guarantee the the similarity of the rest and leads to bias in estimating the treatment effect. To match the generalized propensity scores, we will explore an alternative approach called Vector Matching VM. We are going to discuss some extensions of VM as well.

3.1 Vector Matching

A crucial prerequisite of Vector Matching is to identify the region of common support, which ensures that the matching is made between units who are comparable across treatment groups (Lopez & Gutman, 2017; Scotina & Gutman, 2019). In other words, for any treatments we are analyzing, we want to ensure that there are people in all treatment groups who had a similar propensity scores of that treatment based on their covariates. To do that, we define $r(t, \mathbf{X}|T=t_1)^{(low)}$ as the maximum of the minimum estimated propensity scores for treatment t within each treatment group. That is, we find the minimized $r(t, \mathbf{X})$ for units receiving each treatment respectively, and then find the maximum one. In contrast, $r(t, \mathbf{X}|T=t_1)^{(high)}$ is defined as the minimum of the maximum estimated propensity scores for treatment t t within each treatment group. Mathematically,

$$\begin{aligned} r(t,\mathbf{X})^{(low)} &= \max\left\{\min\left(r\left(t,\mathbf{X} \mid T=t_1\right)\right), \ldots, \min\left(r\left(t,\mathbf{X} \mid T=t_Z\right)\right)\right\} \\ r(t,\mathbf{X})^{(high)} &= \min\left\{\max\left(r\left(t,\mathbf{X} \mid T=t_1\right)\right), \ldots, \max\left(r\left(t,\mathbf{X} \mid T=t_Z\right)\right)\right\}, \end{aligned}$$

and we then select only those units whose estimated GPS values for all treatments fall within the corresponding common support region. We define the indicator variable E_i for each unit i such that $E_i = \mathbf{1}\{r(t,\mathbf{X}) \in \{r(t,\mathbf{X})^{(low)}, r(t,\mathbf{X})^{(high)}\}\ \forall t \in \mathcal{T}\}$, and we only select the units whose $E_i = 1$.

After excluding the units outside the common support region, we would re-fit the GPS model to ensure the comparability across the treatments.

For Vector Matching, the key problem it is trying to solve is how to match units between two treatment groups while ensuring similarity across the entire generalized propensity score (GPS) vector. To solve this, it has two key steps. Suppose we are matching the units from treatment t and t', Vector Matching firstly use clustering methods such as K-means clustering on the rest $Z-2$ components of GPS vector to ensure that the units are similar within each structure. Then it matches units in treatment t and t' within each cluster. These two steps will make sure the global similarity across all GPS components (Lopez & Gutman, 2017; Scotina & Gutman, 2019). The detailed Vector Matching algorithm follows as:

1. Select a reference treatment $t \in \mathcal{T} = \{t_1, t_2, \ldots, t_Z\}$.
2. Compute the GPS vector $R(\mathbf{X}_i)$ for each subject $i = 1, \ldots, N$ using multinomial logistic regression.
3. Define the common support region $\{r(t, \mathbf{X})^{(low)}, r(t, \mathbf{X})^{(high)}\}\ \forall t \in \mathcal{T}$ for multi-treatment propensity scores to ensure overlap across all treatment groups.
4. Remove units with propensity scores that fall outside the common support region. Refit the propensity score model after dropping these units to ensure valid estimates.
5. For each treatment pair (t, t') where $t' \neq t$:
   a. Classify all units using K-means clustering with $K = Z$ based on the logit transform of the rest $Z-2$ components of the GPS vector (excluding $\hat{r}(t, \mathbf{X})$ and $\hat{r}(t', \mathbf{X})$).
   b. Within each cluster, perform 1 to 1 matching between subjects receiving t and t' on $\text{logit}(\hat{r}(t, \mathbf{X}_i))$ with replacement using a caliper of $\epsilon \times \text{sd}\left(\text{logit}\left(\hat{r}(t, \mathbf{X})\right)\right)$, where $\epsilon = 0.2$. The caliper here is the maximum distance we can accept between the units.
6. Units from the treatment are matched to each of the $Z-1$ treatments, and form the final cohort matched data sets.

It is important to emphasize that VM does not produce a single combined matched dataset across all treatments. Instead, it yields $Z-1$ separate matched datasets, each containing only the units from the reference treatment t and one of the other $Z-1$ treatments. Each dataset is constructed independently and used to estimate pairwise treatment effects.

3.2 Extensions to Vector Matching

The basic version of Vector Matching (VM) provides an effective way to match units in multi-treatment settings based on their generalized propensity scores (GPS). However, it also has several limitations. First, because VM employs 1:1 nearest-neighbor matching, not all units with similar propensity scores will necessarily be matched. As a result, some samples may be excluded from the matched dataset, leading to efficiency loss and potential bias in treatment effect estimation. Second, recall that VM performs matching based on a single component $\text{logit}(\hat{r}(t, \mathbf{X}_i))$. As the number of treatments increases, it may not be adequate to control the imbalance on the remaining $Z-1$ components of the GPS vector. Although K-means clustering is used to stratify units prior to matching, clustering does not guarantee to balance the covariates. Third, K-means clustering introduces rigid boundaries in the covariate space. As a results, units located near the edges of clusters may fail to find suitable matches. This limitation becomes more pronounced in high-dimensional GPS, leading to further loss in sample size and an increase in estimation error.

To address the above limitations, Scotina and Gutman (Scotina & Gutman, 2019) proposed some extensions on VM. Firstly, to mitigate the impact of limited matching, we adopt 1:2 matching,

referred to as VM2. Each unit in the treatment t will be matched to two units instead of 1. This will increase the matched sample size, thus improve efficiency while still maintaining covariate balance. In addition, to improve balance across all components of the GPS vector, we replace univariate matching which based solely on $\text{logit}(\hat{r}(t, \mathbf{X}_i))$ with multivariate matching using the Mahalanobis distance. More specifically, the matching is performed based on $(\text{logit}(\hat{r}(t, \mathbf{X})), \text{logit}(\hat{r}(t, \mathbf{X})))'$, referred as VM_MD. We will compare the performance between the two extensions and the original VM through a simulation study on the next section.

4 Simulations

In much of the existing literature on matching based on generalized propensity scores (GPS), simulation studies primarily focus on evaluating covariate balance across treatment groups after matching (Lopez & Gutman, 2017; Scotina & Gutman, 2019). These studies typically simulate covariates with different conditional distributions $\mathbf{X} \mid T = t$ and see the balance of each covariates after the matching. However, they disregard the most important thing in causal inference, which is to estimate ATE or ATT from the matched data sets. Therefore, the simulation study in this paper would focus on two things. First, we will still illustrate the balance of covariates after the matching by using some metrics introduced later. Second, and more importantly, we will evaluate the accuracy of treatment effect estimation by comparing the estimated ATEs from the matched datasets to the true treatment effects specified in the data-generating process. The matching methods we will employ are the Vector Matching VM, Vector Matching with two matches for each unit VM2 and Vector Matching using Mahalanobis distance VM_MD.

4.1 Simulation Metrics

As this simulation study has two main objectives on 1) checking the balance of covariates after matching and 2) estimate the ATE from the matched data. We will adopt the following metrics to asses the performance of each method.

To compare the balance of covariates, we follow the metric propsed by Lopez and Gutman, which employs weighted mean to check the balance (Lopez & Gutman, 2017). For each of the $Z - 1$ matched data set, let n_{pair} be the number of pairs in each matched data, and let ψ_i be the number of times of unit i that in the pairs. The weighted mean of X_p for treatment t where $p \in \{1, \ldots, P\}$, $t \in \{1, \ldots, T\}$, $\bar{X}_{pt}$ is defined as

$$\bar{X}_{pt} = \frac{\sum_{i=1}^{N} X_{pi} \mathbf{I}_i(t) \psi_i}{n_{trip}}$$

where $\mathbf{I}_i(t)$ is an indicator variable that equals to 1 if and only if the unit i actually receive t. Consequently, within each matched data set, we define the standardized bias for each covariate X_p between the treatment j and k, SB_{pjk}as

$$SB_{pjk} = \frac{\bar{X}_{pj} - \bar{X}_{pk}}{\delta_{pt}}$$

where δ_{pt} is the standard deviation of X_p among the units who receive the reference treatment t. It is possible that t equals to j or k. Rubin and Thomas suggested that a standardized bias below 0.25 is generally considered acceptable for making valid causal inferences (Rubin & Thomas, 1996). In addition on that, a cutoff 0.2 is suggested under the multi-treatment settings (McCaffrey et

al., 2013). To estimate the imbalance across all covariates, Lopez and Gutman (Lopez & Gutman, 2017) further extend this idea by taking the maximum value of standardized pairwise bias for each covariate X_p, denoted by $\text{Max}\, 2SB_p$, defined as

$$\text{Max}\, 2SB_p = \max\left(|SB_{p12}|, |SB_{p13}|, |SB_{p23}|, \ldots\right)$$

where each $|SB_{pij}|$ corresponds to each matched data set. By calculating $\text{Max}\, 2SB_p$ for each covariate, it tells us the largest discrepancy of each covariate across all possible pairs of treatments.

However, a potential issue arises when calculating the $\text{Max}\, 2SB_p$. Since each matched data set only contains units from the reference treatment t and the matched units from only one of the $Z-1$ treatments, not all pairwise comparisons are directly represented. For instance, if treatment 1 is chosen as the reference, we obtain matched datasets for comparisons between treatment 1 and treatments $\{2, 3, ..., Z\}$, but not between treatments 2 and 3. As such, SB_{p23} cannot be computed from the matched datasets with reference group 1 alone. Although this limitation is not explicitly mentioned by Lopez and Gutman (2017), we need to repeat the matching procedure by switching the reference treatment. At the end, we will have p different $\text{Max}\, 2SB_p$ and we will the average of them, $\overline{\text{Max}\, 2SB} = \frac{1}{P}\sum_{p=1}^{P} \text{Max}\, 2SB_p$, to measure the overall balance of the matching.

In addition to evaluating covariate balance, our second goal is to estimate the average treatment effect (ATE) from the matched datasets. However, as previously mentioned, the recent literatures of matching on GPS does not include too much details on estimating the ATE based on the matched data sets. This paper will adopt the waysfrom multi-level treatment, which is slightly different with multi-treatments, to estimate the ATE (Lin et al., 2019). The method assigns inverse probability weights based on the estimated propensity scores. Specifically, for a given treatment. Specifically, given the treatment t, each unit is assigned by a weight $w_i(t)$ where $w_i(t) = \frac{1}{\hat{p}_i(t|x)}$. Based on the assigned weights, the expected outcome for treatment t is

$$\hat{\mu}_t = \frac{\sum_{i=1}^{n} \mathbf{I}(T_i = t) w_i(t) Y_i}{\sum_{i=1}^{n} \mathbf{I}(T_i = t) w_i(t)}$$

and estimate the average treatment effect between treatment t and s by

$$\widehat{ATE}_{ts} = \hat{\mu}_t - \hat{\mu}_s$$

The R code used to implement this estimation method is provided in the Appendix 6.5.9.

4.2 Simulation setup

The simulation design in this paper was adpoted from the design from multi-level treatments settings (Lin et al., 2019), with several modifications tailored to the multi-treatment setting. In this study, we will assume the outcome Y is continuous and there are only three treatments for simplicity. To simulated data sets, we firstly generate two variables X_1, X_2 that are related to both treatment assignment and outcome, which are called as the confounders in our design. Moreover, we generate X_3 that are only associated with the treatment assignment and X_4 that is only related to the outcome. We will assume all variables are continuous and follow normal distribution with mean zero and standard deviation 1. Let the treatment space $\mathcal{T} = \{1, 2, 3\}$ such that $|\mathcal{T}| = 3$. The treatment

assignment probabilities are generated using a multinomial logistic model as follows:

$$P(T_i = 1 \mid \mathbf{X}_i) = \frac{1}{1 + \exp\{f_1(\mathbf{X}_i)\} + \exp\{f_2(\mathbf{X}_i)\}}$$
$$P(T_i = 2 \mid \mathbf{X}_i) = \frac{\exp\{f_1(\mathbf{X}_i)\}}{1 + \exp\{f_1(\mathbf{X}_i)\} + \exp\{f_2(\mathbf{X}_i)\}}$$
$$P(T_i = 3 \mid \mathbf{X}_i) = \frac{\exp\{f_2(\mathbf{X}_i)\}}{1 + \exp\{f_1(\mathbf{X}_i)\} + \exp\{f_2(\mathbf{X}_i)\}}$$

where $f_j(\mathbf{X})$ are functions of covariates $\mathbf{X}_i$. In this study, we consider five different specifications for these functions, including linear, quadratic, interaction, and combination forms (details provided in the Appendix 6.3). In addition, the true data-generating process for the outcome is specified as:

$$\mathbb{E}[Y|\mathbf{X}] = 4 + 0.3X_1 - 0.2X_2 + 0.6X_4 - 0.5\mathbf{I}\{T = 2\} + 0.7\mathbf{I}\{T = 3\}$$

where $\mathbf{I}\{T = j\}$ is an indicator variale that equals to 1 if and only if the treatment received for each unit is j. It can be computed that the true $ATE_{1,2} = 0.5$, $ATE_{1,3} = -0.7$ and $ATE_{3,2} = -1.2$ (see Appendix 6.2).

When performing the Vector Matching, we choose the treatments 1 and 2 as reference treatments in separate iterations. As a result, the VM yields two sets of matched datasets: 1) units receiving treatment 1 matched to those receiving treatments 2 and 3, and 2 units receiving treatment 2 matched to those receiving treatments 3. As illustrate in the previous subsection, overall standardized bias simplifies to:

$$\text{Max}\, 2SB_p = \max\left(|SB_{p12}|, |SB_{p13}|, |SB_{p23}|\right)$$

For each scenario described in Appendix 6.3, we take the sample size to be 3000. The full simulation works as follows:

1. Generate 3000 samples using the given scenario of $f_j(\mathbf{X}_i)$.
2. Compute the generalized propensity scores for each unit using multinomial logistic regression, and compute the common support region described previously.
3. Select the units whose propensity scores satisfies the common support region, and re-fit the model to estimate the GPS again.
4. Choose the reference treatment t as treatment 1. For each $t' \in \{2, 3\}$, we perform the K-means clustering on $l \neq t, t'$, then perform the three matching methods (VM, VM2 and VM_MD) described in Section 3. This will give us matched data sets for pair $(1, 2)$ and $(2, 3)$. Repeat for each matching method and will return six data sets.
5. Repeat 4 but set the reference treatment to be 2. Take only the matched data for pair $(2, 3)$. Repeat for each matching method and will return three data sets.
6. Use the resulting data sets to compute the $\overline{\text{Max}\, 2SB}$ and matched rate which is the proprotion of units being matched across all matched data sets. We will also estimate the two average effect $\widehat{ATE_{12}}$ and $\widehat{ATE_{13}}$.
7. Repeat Steps 1 to 5 for 300 times.

While we can also estimate $\widehat{ATE_{23}}$, we focus on treatment pairs involving the same reference group (i.e., treatment 1), to allow for more consistent estimates across the methods. The codes of generating the data can be found at Appendix 6.5.1. Appendix 6.5.2 to 6.5.9 shows the functions of steps 2 to 5. Appendix 6.5.10 shows the codes for running one simulation and Appendix 6.5.11 shows how to fun the simulation for multiple times.

4.3 Simulation results

Due to the space limit, all the plots that are described at the main text are moved to Appendix 6.4.1.

Figure 1 presents the distribution of the four covariates across the three treatment groups in a single simulated dataset under Scenario 1 in Appendix 6.3. It shows notable discrepancies in covariate distributions between treatment groups, with particularly large differences observed in X_1, X_2 and X_3. This imbalance underscores a key concern in causal inference that the direct estimation of ATE without adjustment for confounding variables would likely result in substantial bias and misleading conclusions. This also highlights the necessarity of methods such as propensity score matching.

Following the matching steps, Figure 2 shows the distribution of $\overline{\text{Max}\, 2SB}$ and the matching rate under the four methods under scenario 1 in Appendix 6,3. We can see that all three method all has a distribution with mean bias larger than the cutoff 0.2 suggested by the McCaffrey et al. (McCaffrey et al., 2013), suggesting that none of these method eliminate the bias perfectly. Among the three matching methods, the Vector Matching with two matches has the highest bias, whose mean bias is around 2. Followed by VM2, the original matching has the medium bias with mean around 1.5. In addition, the Vector Matching using Mahalanobis distance generates the lowest bias, with main part lies between 0.7 and 0.8. All three methods have relatively good performance on matching rate, which are all above 80%. In addition, the pattern observed in the matching rate mirrors that of the covariate bias. Methods that yield higher bias tend to retain a larger proportion of matched units, while those that enforce stricter similarity criteria result in lower matching rates. This outcome is expected, as the use of Mahalanobis distance imposes more stringent matching constraints, leading to a lower matching rate.

In addition to the bias and matching rate, Figure 3 shows the distribution of estimation of the two treatment effects under the Scenario 1 in Appendix 6.3. Recall that the true treatment effects are 0.2 and -0.7. All three methods seems to have a relatively same performance, where the mean estimations for the two are around 0.55 and -0.8. However, it seems that the results using VM_MD exhibit slightly greater deviation from the true values compared to the other methods. The reason is might be the lower matching rate of VM_MD, which leads to a loss of sample size and reduced data variability, thereby increasing the variance of the estimates. Furthermore, we can observe the similar patterns as shown in Figure 1 to 3 for other scenarios in Appendix 6.3, indicating a consistent performance under different treatment assignment.

Finally, Figure 4 compares the estimation of the two treatment effects under the five scenarios of f_j under the three methods. For the estimation of ATE_{12}, the VM and VM2 seems to have the similar performance, where their estimates are similar under each scenarios. In contrast, VM_MD seems to generate the worse estimates, which has an approximate 0.02 estimation gap between the other two methods. Interestingly, all three methods have perfect estimation in Scenario 2, where the estimates are both very close to the true value 0.5. A similar pattern is observed for the estimation of ATE_{13}, where VM_MD only generates the similar estimates as the other two in scenarios 2 and 3, and has notable discrepancies on the rest. Furthermore, all three methods perform less accurately in estimating ATE_{13} compared ATE_{12}, indicating that the estimation of treatment effects involving treatment 3 may be more sensitive to imbalance or model misspecification.

5 Discussion

This paper introduced the generalized propensity scores in multi-treatment settings, and discuss the propensity score matching for generalized propensity scores, Vector Matching. This paper has conducted an simulation study to illustrate the performance of each matching method using different data-generating scheme. Overall, the performance of the Vector Matching and its extensions are satisfied. From Figure 2 to 4, while the standardized bias remains relatively high, the matching rate are consistently high and the estimation of ATEs generally close to the true values. In addition, from Figure 4, all three methods has good performance under different treatment assignment. These findings underscore the effectiveness of Vector Matching.

However, the recent study on propensity scores focus more on how to estimate them instead of how to use them. In practice, alternative methods such as Inverse Probability Weighting or Doubly Robust Estimators (Li & Li, 2019), or propensity score adjustment (Feng et al., 2012), are more widely accepted due to the ease of implementation. There are several reasons on limiting use of matching methods. Firstly, this method is too computational intensive, especially when the number of treatments and number of covariates increases. In this simulation study, we considered a simplified setting with only three treatment groups and conducted 300 iterations, which is fewer than the common simulation study. This is due to the complexity and size of the function used to run a single simulation iteration (see Appendix 6.5.10). Even under the simplest scenario (Scenario 1), running 300 simulations required approximately five minutes. The computation time increases significantly when quadratic or interaction terms are introduced into the data-generating process. As a results, increasing the number of treatments will significantly increase the computational burden. This is the primary reason why the current study is limited to three-treatment settings.

As a consequence of the relatively low number of simulation iterations, the results presented in Figures 2 and 3 may not reflect the asymptotic performance of Vector Matching. If we can increase the number of simulation to, say 1000 times, then the estimated imbalance across the covariates should be lower, and the results of estimation of the two ATEs should be closer to the true treatment effects.This suggests that, with a sufficiently large number of iterations, Vector Matching has the potential to achieve the cutoff suggest by Rubin and Thomas (1996) and McCaffrey et al. (2013). Interestingly, even under the current limited number of simulations, Figure 4 shows a notable difference in performance between the estimation of ATE_{12} and ATE_{13}, despite they all using treatment 1 as the reference. A plausible explanation for this discrepancy is that the coefficients in $f_1(\mathbf{X})$ are moderate and aligned in the same direction. In addition, in the outcome model, the treatment effect for treatment 3 (+0.7) is larger than that for treatment 2 (-0.5). This may increase the residual unbalance when estimating ATE_{13} to be 1.4 timer higher than ATE_{12}.

Future direction should be mainly concentrate on two things. Firstly, there are other possible extensions on Vector Matching except the two listed on this paper (Scotina & Gutman, 2019). For example, one possible way is to use the fuzzy clustering instead of K-means clustering in step 5 a in Section 3.1, which can provides better classification abilities on the boundries of clustering. Another one is that we may not set the caliper in both original VM or VM using Mahalanobis distance. Without a caliper, the algorithm becomes more flexible by allowing greater variation in the matched distances between units. Interestingly, the original VM without the caliper seems to have the best performance on reducing the bias (Scotina & Gutman, 2019). Finally, the performance of Vector Matching when the number of treatments and covariates is large relative to the sample size remains unclear. This would be a challenging topic as it combines small-sample inference and high-dimensional matching, which needs further research.

6 Appendix

6.1 Proof of balancing property and exchangebility of propensity scores

We will prove under binary treatment setting, the proof in multi-treatments settings is similar.

$r(t, \mathbf{X})$ is called the balancing score if

$$T \perp \mathbf{X} \mid r(t, \mathbf{X})$$

proof: We know that

$$\begin{aligned} P[T = 1 \mid r(t, \mathbf{X}), L] &= P[T = 1 \mid L] \\ &= r(t, \mathbf{X}) \end{aligned}$$

This imples that

$$\begin{aligned} P[T = 1 \mid r(t, \mathbf{X})] &= E[T \mid r(t, \mathbf{X})] \\ &= E\{E[T \mid r(t, \mathbf{X}), L] \mid r(t, \mathbf{X})\} \\ &= E\{E[T \mid L] \mid r(t, \mathbf{X})\} \\ &= E\{r(t, \mathbf{X}) \mid r(t, \mathbf{X})\} \\ &= r(t, \mathbf{X}) \end{aligned}$$

This implies

$$P[T = 1 \mid r(t, \mathbf{X}), L] = P[T = 1 \mid r(t, \mathbf{X})]$$

Equivalently

$$T \perp L \mid r(t, \mathbf{X})$$

■

Next, we want to show the exchangebility of propensity scores, that is

$$\left(Y^0, Y^1\right) \perp T \mid r(t, \mathbf{X})$$

proof:

$$\begin{aligned} P\left[T = 1 \mid Y^0, Y^1, r(t, \mathbf{X})\right] &= E\left[T \mid Y^0, Y^1, r(t, \mathbf{X})\right] \\ &= E\left[E\left[T \mid L, Y^0, Y^1, r(t, \mathbf{X})\right] \mid Y^0, Y^1, r(t, \mathbf{X})\right] \\ &= E\left[E\left[T \mid L, Y^0, Y^1\right] \mid Y^0, Y^1, r(t, \mathbf{X})\right] \\ &= E\left[E[T \mid L] \mid Y^0, Y^1, r(t, \mathbf{X})\right] \\ &= E\left[r(t, \mathbf{X}) \mid Y^0, Y^1, r(t, \mathbf{X})\right] \\ &= r(t, \mathbf{X}) \\ &= P[T = 1 \mid r(t, \mathbf{X})] \end{aligned}$$

Thus it implies

$$\left(Y^0, Y^1\right) \perp A \mid r(t, \mathbf{X})$$

■

6.2 Compute the true ATE

Given the true model generating the outcome is

$$\mathbb{E}[Y|\mathbf{X}] = 4 + 0.3X_1 - 0.2X_2 + 0.6X_4 - 0.5\mathbf{I}\{T = 2\} + 0.7\mathbf{I}\{T = 3\}$$

The ATE_{12} is the average difference between the potential outcome of receiving 1 and 2. Recall the assumption of exchangebility and consistency we introduced initially, ATE_{12} can be expressed as

$$\begin{aligned} ATE_{12} &= E[Y^1] - E[Y^2] \\ &= E[Y^1 \mid T = 1] - E[Y^2 \mid T = 2] \quad \text{by exchangebility} \\ &= E[Y \mid T = 1] - E[Y \mid T = 2] \quad \text{by consistency} \\ &= 0 - (-0.5) \quad \text{from the funtion of generating Y} \\ &= 0.5 \end{aligned}$$

Similarly, we can calculate $ATE_{13} = -0.7$ and $ATE_{23} = -1.2$

6.3 The list of scanerios

Case 1:

$$\begin{aligned} f_1 &= 0.5X_1 + 0.3X_2 + 0.4X_3 \\ f_2 &= -0.2X_1 + 0.6X_2 - 0.3X_3 \end{aligned}$$

Case 2:

$$\begin{aligned} f_1 &= 0.5X_1 + 0.3X_2 + 0.4X_3 - 0.4X_1^2 \\ f_2 &= -0.2X_1 + 0.6X_2 - 0.3X_3 + 0.3X_2^2 \end{aligned}$$

Case 3:

$$\begin{aligned} f_1 &= 0.5X_1 + 0.3X_2 + 0.4X_3 - 0.4X_1^2 + 0.3X_2^2 \\ f_2 &= -0.2X_1 + 0.6X_2 - 0.3X_3 + 0.3X_2^2 + 0.2X_3^2 \end{aligned}$$

Case 4:

$$\begin{aligned} f_1 &= 0.5X_1 + 0.3X_2 + 0.4X_3 - 0.2X_1X_2 \\ f_2 &= -0.2X_1 + 0.6X_2 - 0.3X_3 + 0.1X_2X_3 \end{aligned}$$

Case 5:

$$\begin{aligned} f_1 &= 0.5X_1 + 0.3X_2 + 0.4X_3 - 0.4X_1^2 - 0.2X_1X_2 \\ f_2 &= -0.2X_1 + 0.6X_2 - 0.3X_3 + 0.3X_2^2 + 0.1X_2X_3 \end{aligned}$$

6.4 Graphics

This section contains all graphics that are important in this study.

6.4.1 Graph that should be included on the text but lack of space

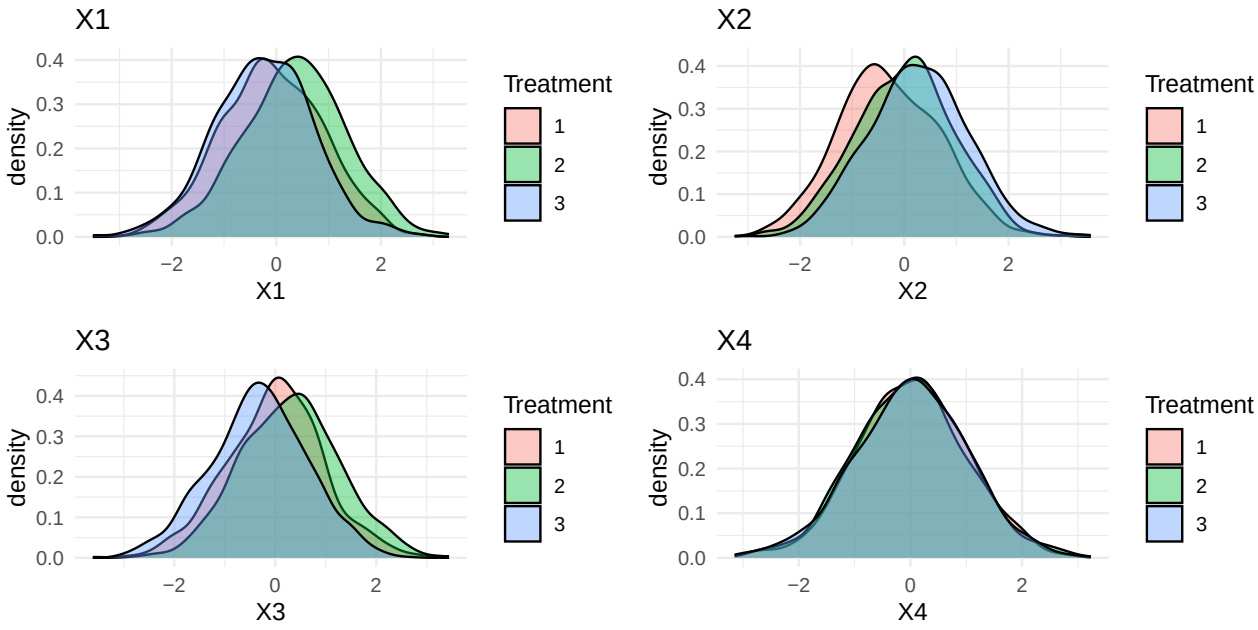

Figure 1: *Covariate density by treatment group for one simulation before matching*

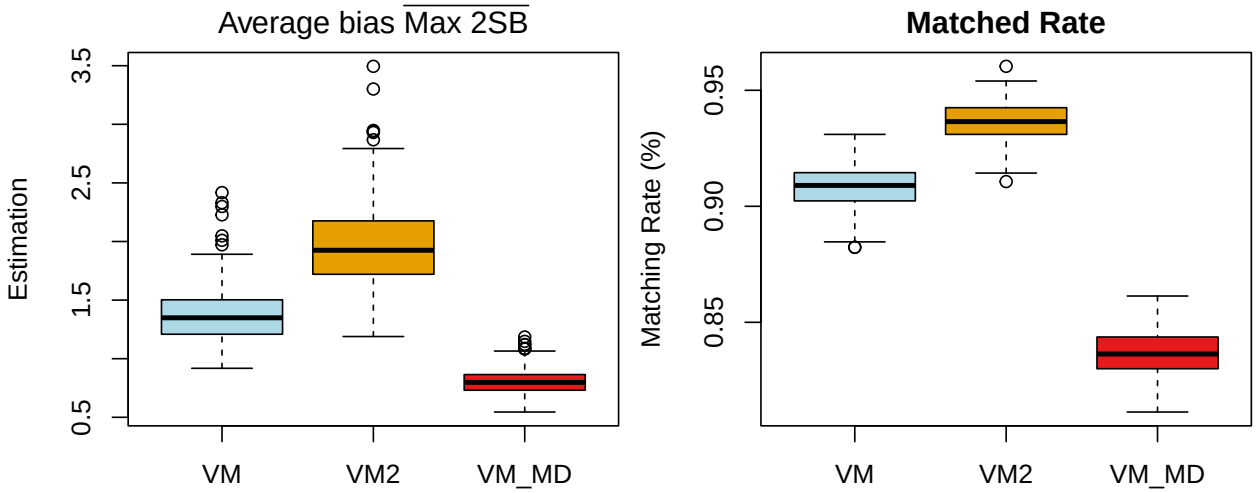

Figure 2: *Average of maximum pair-wise treatment bias and corresponding matching rate under each method (Scenario 1).*

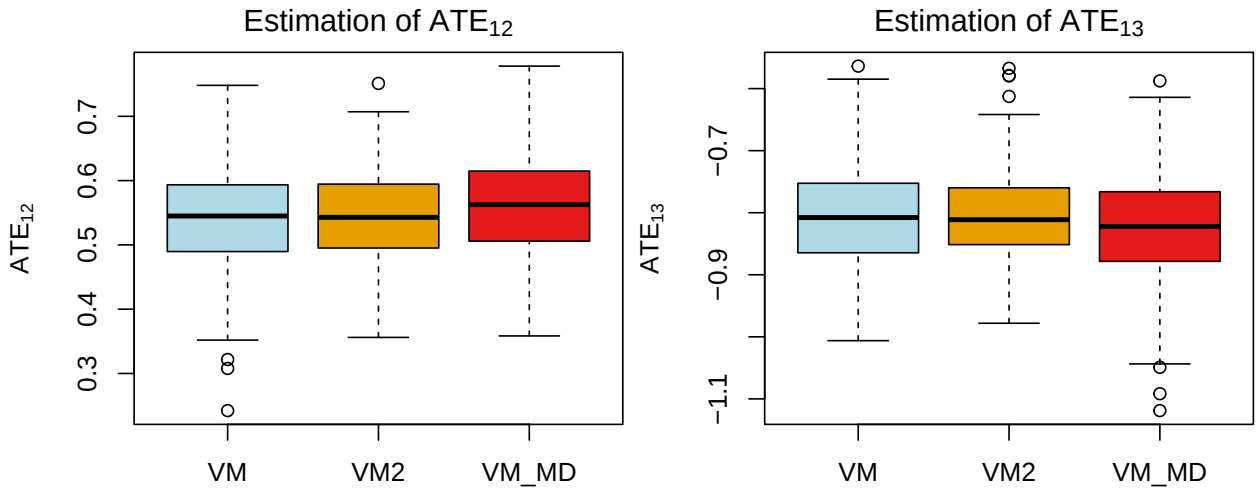

Figure 3: *Distributions of ATE Estimates for Treatments 1 vs 2 and 1 vs 3 under each method (Scenario 1).*

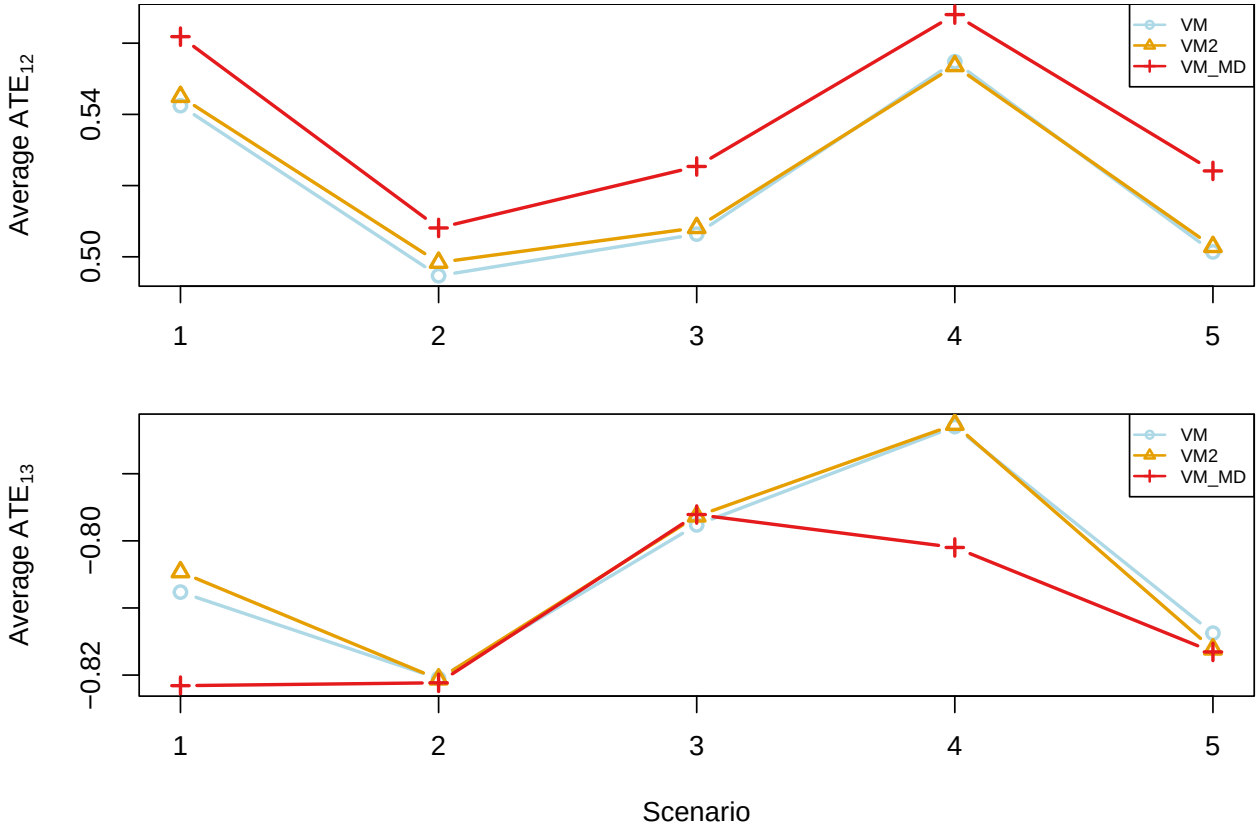

Figure 4: *Comparision of estimation on two treatment effects under different scenarios under different methods.*

6.4.2 Graphics for other scenarios discibed in 6.3

The average bias, matching rate estimation of the two treatment effect for the rest four scenarios that we do not have the space to put them on the main text.

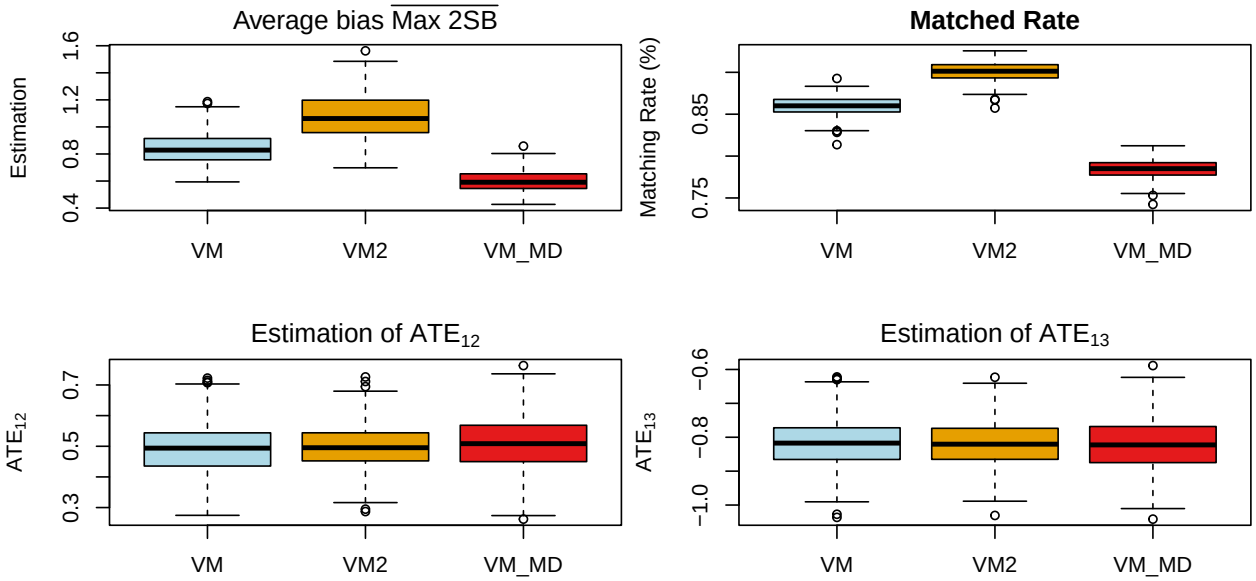

Figure 5: *The average bias, matching rate and estimation of the two treatment effect under scenario 2*

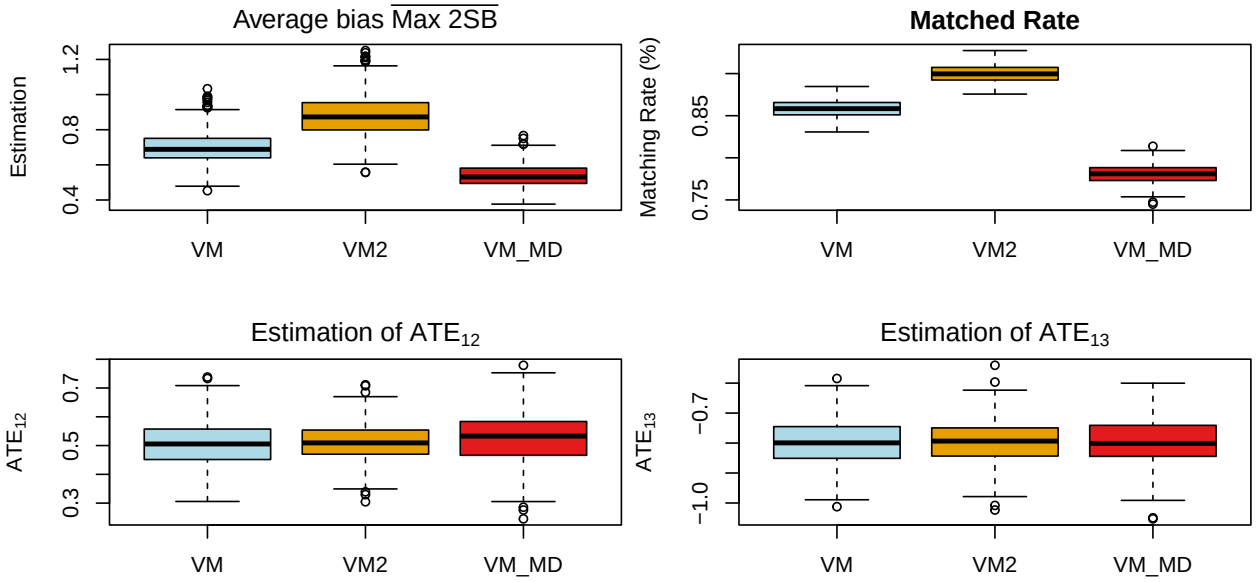

Figure 6: *The average bias, matching rate and estimation of the two treatment effect under scenario 3*

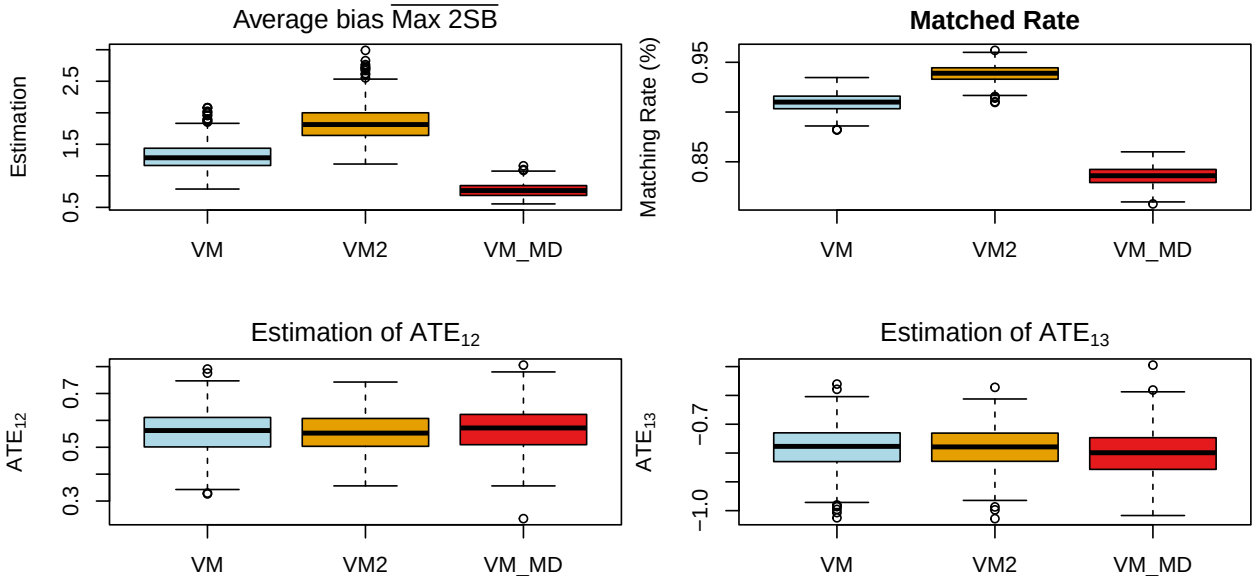

Figure 7: *The average bias, matching rate and estimation of the two treatment effect under scenario 4*

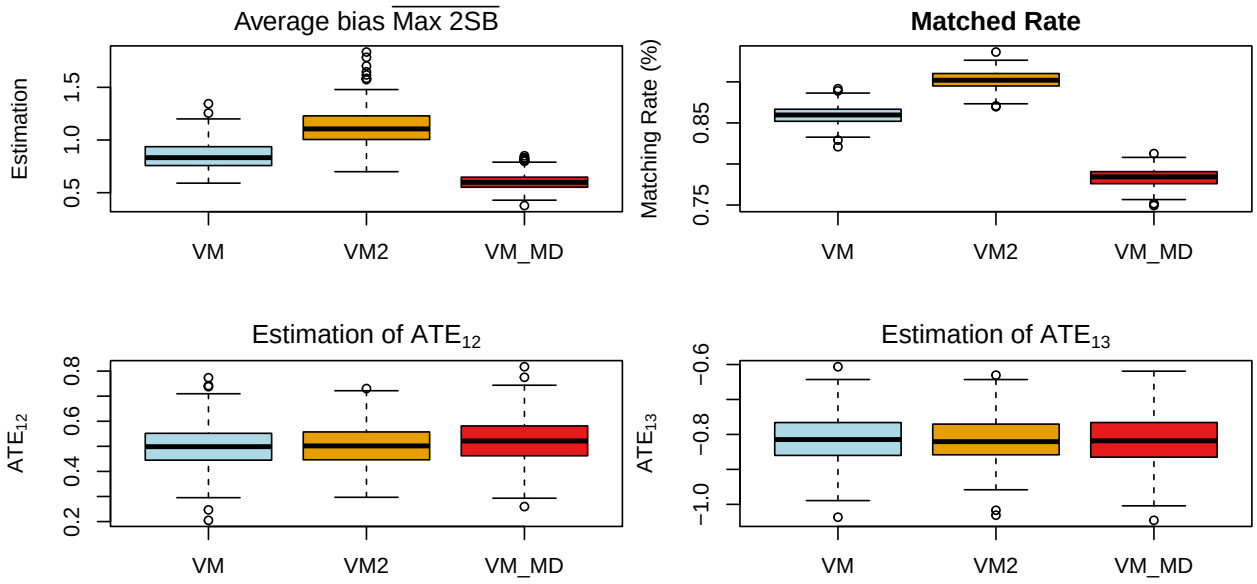

Figure 8: *The average bias, matching rate and estimation of the two treatment effect under scenario 5*

6.5 R codes

6.5.1 Generating the data set

The R function that generating the data

```r
# Helper function to generate the data under Scenario 1
simulate_multitreatment_data <- function(n = 3000, seed = 123) {
  set.seed(seed)

  # Generate covariates
  X1 <- rnorm(n, mean = 0, sd = 1)
  X2 <- rnorm(n, mean = 0, sd = 1)
  X3 <- rnorm(n, mean = 0, sd = 1)
  X4 <- rnorm(n, mean = 0, sd = 1)

  # Define linear predictors for the multinomial logistic
  f1 <-  0.5 * X1 + 0.3 * X2 + 0.4 * X3
  f2 <- -0.2 * X1 + 0.6 * X2 - 0.3 * X3

  # Multinomial probabilities
  denom <- 1 + exp(f1) + exp(f2)
  p1 <- 1       / denom
  p2 <- exp(f1) / denom
  p3 <- exp(f2) / denom

  # Draw treatment T
  T <- numeric(n)
  for (i in seq_len(n)) {
```

```
    T[i] <- sample(x = 1:3, size = 1, prob = c(p1[i], p2[i], p3[i]))
  }

  # Outcome model + noise
  base_mean <- 4 + 0.3 * X1 - 0.2 * X2 + 0.6 * X4
  Y <- base_mean +
    ifelse(T == 2, -0.5, 0) +
    ifelse(T == 3,  0.7, 0) +
    rnorm(n, mean = 0, sd = 1)

  return(data.frame(id = 1:n, X1 = X1, X2 = X2, X3 = X3, X4 = X4,
                    Treatment = T, Y = Y))
}
```

6.5.2 Estimate the Generalized Propensity Scores

The R function that estimate the generalized propensity scores using multinomial logistic regressions.

```
library(nnet)  # for multinom
estimate_propensity_scores <- function(data) {
  data$Treatment <- as.factor(data$Treatment)
  model <- multinom(Treatment ~ . - Treatment - id - Y, data = data,
                    trace = FALSE)
  propensity_scores <- predict(model, type = "probs")
  treatment_levels <- levels(data$Treatment)
  colnames(propensity_scores) <- paste0("PS", treatment_levels)
  return(propensity_scores)
}
```

6.5.3 Compute the common support region

The R function that compute the common support region and filtered the valid index.

```
common_support <- function(ps_matrix, treatment_col) {
  treatment_groups <- unique(ps_matrix[[treatment_col]])
  low_values  <- numeric(length(treatment_groups))
  high_values <- numeric(length(treatment_groups))
  for (target_treatment in treatment_groups) {
    ps_target <- ps_matrix[[paste0("PS", target_treatment)]]
    group_min <- sapply(treatment_groups, function(actual_treatment) {
      group_data <- ps_matrix[ps_matrix[[treatment_col]] == actual_treatment, ]
      ps_values <- group_data[[paste0("PS", target_treatment)]]
      return(min(ps_values))
    })
    group_max <- sapply(treatment_groups, function(actual_treatment) {
      group_data <- ps_matrix[ps_matrix[[treatment_col]] == actual_treatment, ]
      ps_values <- group_data[[paste0("PS", target_treatment)]]
      return(max(ps_values))
    })
```

```
    low_values[target_treatment]  <- max(group_min)
    high_values[target_treatment] <- min(group_max)
  }
  return(list(Low  = low_values,
              High = high_values))
}

filter_valid_subjects_indices <- function(ps_data, css) {
  low  <- css$Low
  high <- css$High
  is_valid <- apply(ps_data, 1, function(row) {
    all(row >= low & row <= high)
  })
  return(which(is_valid))
}
```

6.5.4 Perform the K-means clustering

The R function that perform the K-means clustering based on the l such that $l \neq t, t'$.

```
perform_kmc <- function(ps_data, ref_treatment, t_prime, K = 5) {
  # Filter the propensity scores
  ps_subset <- ps_data[, !colnames(ps_data) %in% c(ref_treatment, t_prime),
                       drop = FALSE]

  # Apply logit transformation to the remaining propensity scores
  logit_transform <- function(p) log(p / (1 - p))
  ps_logit <- apply(ps_subset, 2, logit_transform)

  # Perform KMC with the specified K
  set.seed(123)
  final_kmc <- kmeans(ps_logit, centers = K, nstart = 20)

  return(list(clusters = final_kmc$cluster,
              kmc_model = final_kmc))
}
```

6.5.5 Vector Matching VM

```
library(MatchIt)
library(Matching)

perform_matching <- function(data, ref_treatment, t_prime, epsilon = 0.2) {
  all_matches <- data.frame()
  unique_clusters <- unique(data$Cluster)

  for (k in unique_clusters) {
    cluster_data <- data[data$Cluster == k, ]
```

```
    # logit(PS(ref_treatment))
    logit_transform <- function(p) log(p / (1 - p))
    cluster_data$logit_r_t <- logit_transform(cluster_data[[paste0("PS", ref_treatment)]])

    # Keep T in {ref_treatment, t_prime}
    cluster_data$treatment_indicator <- ifelse(
      cluster_data$Treatment == ref_treatment, 1,
      ifelse(cluster_data$Treatment == t_prime, 0, NA)
    )
    cluster_data <- subset(cluster_data, !is.na(treatment_indicator))
    if (nrow(cluster_data) < 2) next

    caliper_width <- epsilon * sd(cluster_data$logit_r_t, na.rm = TRUE)
    X  <- cluster_data$logit_r_t
    Tr <- cluster_data$treatment_indicator

    match_result <- Match(
      Y = NULL,
      Tr = Tr,
      X = X,
      M = 1,
      caliper = caliper_width,
      replace = TRUE
    )
    matched_id <- data.frame(
      id_ref     = cluster_data$id[ match_result$index.treated ],
      t_ref      = cluster_data$Treatment[ match_result$index.treated ],
      id_t_prime = cluster_data$id[ match_result$index.control ],
      t_prime    = cluster_data$Treatment[ match_result$index.control ]
    )
    all_matches <- rbind(all_matches, matched_id)
  }
  return(all_matches)
}
```

6.5.6 Vector Matching with two matches VM2

```
perform_matching_2 <- function(data, ref_treatment, t_prime, epsilon = 0.2,
                               n_match = 2) {
  all_matches <- data.frame()
  unique_clusters <- unique(data$Cluster)

  for (k in unique_clusters) {
    cluster_data <- data[data$Cluster == k, ]

    logit_transform <- function(p) log(p / (1 - p))
```

```
    cluster_data$logit_r_t <- logit_transform(cluster_data[[paste0("PS", ref_treatment)]])

    cluster_data$treatment_indicator <- ifelse(
      cluster_data$Treatment == ref_treatment, 1,
      ifelse(cluster_data$Treatment == t_prime, 0, NA)
    )
    cluster_data <- subset(cluster_data, !is.na(treatment_indicator))
    if (nrow(cluster_data) < 2) next

    caliper_width <- epsilon * sd(cluster_data$logit_r_t, na.rm = TRUE)
    X  <- cluster_data$logit_r_t
    Tr <- cluster_data$treatment_indicator

    match_result <- Match(
      Y = NULL,
      Tr = Tr,
      X = X,
      M = n_match,
      caliper = caliper_width,
      replace = TRUE
    )
    matched_id <- data.frame(
      id_ref     = cluster_data$id[ match_result$index.treated ],
      t_ref      = cluster_data$Treatment[ match_result$index.treated ],
      id_t_prime = cluster_data$id[ match_result$index.control ],
      t_prime    = cluster_data$Treatment[ match_result$index.control ]
    )
    all_matches <- rbind(all_matches, matched_id)
  }
  return(all_matches)
}
```

6.5.7 Vector Matching using Mahalanobis distance

```
perform_matching_vm_md <- function(data, ref_treatment, t_prime, epsilon = 0.2) {
  all_matches <- data.frame()
  unique_clusters <- unique(data$Cluster)

  for (k in unique_clusters) {
    cluster_data <- data[data$Cluster == k, ]

    logit_transform <- function(p) log(p / (1 - p))
    cluster_data$logit_r_t <- logit_transform(cluster_data[[paste0("PS", ref_treatment)]])
    cluster_data$logit_r_tprime <- logit_transform(cluster_data[[paste0("PS", t_prime)]])

    cluster_data$treatment_indicator <- ifelse(
      cluster_data$Treatment == ref_treatment, 1,
```

```
      ifelse(cluster_data$Treatment == t_prime, 0, NA)
    )
    cluster_data <- subset(cluster_data, !is.na(treatment_indicator))
    if (nrow(cluster_data) < 2) next

    mat_2d <- as.matrix(cluster_data[, c("logit_r_t", "logit_r_tprime")])

    Sigma <- cov(mat_2d)
    Sigma_inv <- solve(Sigma)
    L <- chol(Sigma_inv)
    WeightedX <- t(L %*% t(mat_2d))

    # caliper in WeightedX space
    WeightedX_mean <- apply(WeightedX, 1, function(z) sqrt(sum(z^2)))
    caliper_width <- epsilon * sd(WeightedX_mean, na.rm = TRUE)

    match_result <- Match(
      Y = NULL,
      Tr = cluster_data$treatment_indicator,
      X = WeightedX,
      M = 1,
      caliper = caliper_width,
      replace = TRUE
    )

    matched_id <- data.frame(
      id_ref     = cluster_data$id[ match_result$index.treated ],
      t_ref      = cluster_data$Treatment[ match_result$index.treated ],
      id_t_prime = cluster_data$id[ match_result$index.control ],
      t_prime    = cluster_data$Treatment[ match_result$index.control ],
      cluster    = k
    )
    all_matches <- rbind(all_matches, matched_id)
  }
  return(all_matches)
}
```

6.5.8 Calculate the pair-wise standardized bias

```
calc_standardized_bias <- function(data, covariates, ref_treatment = 1) {
  # This function expects data with columns:
  #   - id
  #   - Treatment
  #   - pair (or matched set index)
  #   - the covariates in 'covariates'
  #
  # Weighted approach: each subject i has frequency psi_i (# times matched).
```

```r
n_pairs <- length(unique(data$pair))
count_by_id <- table(data$id)
psi_lookup  <- setNames(as.numeric(count_by_id), names(count_by_id))
get_psi <- function(id_i) psi_lookup[as.character(id_i)]

trt_levels <- sort(unique(data$Treatment))
if (length(trt_levels) != 2) {
  stop("Data must have exactly 2 treatments for this SB calculation!")
}

# Weighted means
wmean_mat <- matrix(
  NA,
  nrow = length(covariates),
  ncol = 2,
  dimnames = list(covariates, paste0("T", trt_levels))
)

for (p in seq_along(covariates)) {
  cov_name <- covariates[p]
  for (trt in trt_levels) {
    rows_trt <- which(data$Treatment == trt)
    if (length(rows_trt) == 0) {
      wmean_mat[p, paste0("T", trt)] <- NA
      next
    }
    Xp_vals  <- data[[cov_name]][rows_trt]
    psi_vals <- sapply(data$id[rows_trt], get_psi)
    weighted_sum <- sum(Xp_vals * psi_vals)
    wmean_mat[p, paste0("T", trt)] <- weighted_sum / n_pairs
  }
}

# Weighted SD in the reference group
wsd_vec <- numeric(length(covariates))
names(wsd_vec) <- covariates

ref_col <- paste0("T", ref_treatment)
rows_ref <- which(data$Treatment == ref_treatment)
psi_vals_ref <- sapply(data$id[rows_ref], get_psi)

for (p in seq_along(covariates)) {
  Xp_vals_ref <- data[[covariates[p]]][rows_ref]
  wmean_ref   <- wmean_mat[p, ref_col]

  var_num <- sum(psi_vals_ref * (Xp_vals_ref - wmean_ref)^2)
  var_den <- sum(psi_vals_ref)
```

```
    if (var_den > 1e-10) {
      wsd_vec[p] <- sqrt(var_num / var_den)
    } else {
      wsd_vec[p] <- NA
    }
  }

  # Standardized difference for the other group vs. reference
  other_treatment <- setdiff(trt_levels, ref_treatment)
  other_col <- paste0("T", other_treatment)

  sb_vec <- numeric(length(covariates))
  names(sb_vec) <- covariates
  for (p in seq_along(covariates)) {
    denom <- wsd_vec[p]
    if (is.na(denom) || denom < 1e-10) {
      sb_vec[p] <- NA
    } else {
      mean_other <- wmean_mat[p, other_col]
      mean_ref   <- wmean_mat[p, ref_col]
      sb_vec[p]  <- (mean_other - mean_ref) / denom
    }
  }

  abs_sb <- abs(sb_vec)
  return(abs_sb)
}
```

6.5.9 Estimate the ATE from the matched data set

```
calc_ate_ipw <- function(data, t1 = 1, t2 = 2,
                         y_col = "Y",
                         trt_col = "Treatment",
                         ps_cols = c("PS1", "PS2", "PS3")) {
  data$ipw <- apply(data, 1, function(row) {
    trt_val <- as.numeric(row[[trt_col]])
    if (trt_val == 1) {
      return(1 / as.numeric(row[[ ps_cols[1] ]]))  # 1/PS1
    } else if (trt_val == 2) {
      return(1 / as.numeric(row[[ ps_cols[2] ]]))  # 1/PS2
    } else if (trt_val == 3) {
      return(1 / as.numeric(row[[ ps_cols[3] ]]))  # 1/PS3
    } else {
      return(NA)
    }
  })
```

```r
  # Weighted mean function
  wmean <- function(x, w) sum(x * w) / sum(w)
  # Subset for t1
  idx_t1 <- data[[trt_col]] == t1
  mu_t1 <- wmean(
    x = data[[y_col]][idx_t1],
    w = data$ipw[idx_t1]
  )
  # Subset for t2
  idx_t2 <- data[[trt_col]] == t2
  mu_t2 <- wmean(
    x = data[[y_col]][idx_t2],
    w = data$ipw[idx_t2]
  )
  ate_t1_t2 <- mu_t1 - mu_t2
  return(list(mu_t1 = mu_t1,
              mu_t2 = mu_t2,
              ATE_t1_t2 = ate_t1_t2))
}
```

6.5.10 Run one simulation

The R function to run one simulation using all the function we defined previously. It is a big function.

```r
library(dplyr)

one_simulation <- function(n = 3000,K = 3,
                           matching_method = c("M1", "M2", "VM_MD"),
                           epsilon = 0.2,   n_match = 2, seed = NULL,
                           covariates_for_SB = c("X1","X2","X3","X4")) {
  if(!is.null(seed)) set.seed(seed)

  ## 1) Simulate data
  data <- simulate_multitreatment_data(n = n)

  ## 2) Estimate PS & drop out-of-support
  ps <- estimate_propensity_scores(data)
  ps_trt <- as.data.frame(cbind(ps, Treatment = data$Treatment))
  css <- common_support(ps_trt, "Treatment")
  valid_index <- filter_valid_subjects_indices(ps, css)
  valid_data  <- data[valid_index,]

  ps_new <- estimate_propensity_scores(valid_data)
  valid_data_ps <- cbind(valid_data, ps_new)

  ## 3) For each pair of treatments (1 vs 2), (1 vs 3), (2 vs 3):
  ##    (a) KMC => cluster assignments
```

```r
##    (b) call the chosen matching function => matched pairs
##    (c) build final matched cohort => compute standardized biases

# Helper: choose the correct matching function
run_matching_func <- function(method, data_input, ref_t, t_prime_t) {
  if (method == "M1") {
    out <- perform_matching(data = data_input,
                            ref_treatment = ref_t,
                            t_prime= t_prime_t,
                            epsilon= epsilon)
    }
  else if (method == "M2") {
    out <- perform_matching_2(data = data_input,
                              ref_treatment = ref_t,
                              t_prime = t_prime_t,
                              epsilon = epsilon,
                              n_match = n_match)}
  else if (method == "VM_MD") {
    out <- perform_matching_vm_md(data = data_input,
                                  ref_treatment = ref_t,
                                  t_prime = t_prime_t,
                                  epsilon = epsilon)
  }
  else if (method == "VM_MDnc") {
    out <- perform_matching_vm_mdnc(data = data_input,
                                    ref_treatment = ref_t,
                                    t_prime = t_prime_t)
  } else {
    stop("Unrecognized matching method!")
  }
  return(out)
}

# Pair (1 vs 2)
kmc_12 <- perform_kmc(ps_data = ps_new,
                      ref_treatment = "PS1",
                      t_prime = "PS2",
                      K = K)
valid_data_ps_12 <- cbind(valid_data_ps,
                          Cluster = kmc_12$clusters)
matches_1_2 <- run_matching_func(matching_method,
                                 valid_data_ps_12,
                                 ref_t=1, t_prime=2)

final_cohort_1_2 <- data.frame(id = c(rbind(matches_1_2$id_ref,
                                            matches_1_2$id_t_prime)),
                               t = c(rbind(matches_1_2$t_ref,
```

```
                                          matches_1_2$t_prime )),
                                pair = rep(seq_len(nrow(matches_1_2)),
                                           each = 2))
final_cohort_1_2 <- dplyr::left_join(final_cohort_1_2,
                                     valid_data_ps,
                                     by = c("id", "t" = "Treatment")) %>%
  dplyr::rename(Treatment = t)

abs_sb_12 <- calc_standardized_bias(data = final_cohort_1_2,
                                    covariates = covariates_for_SB,
                                    ref_treatment = 1)
############
# Pair (1 vs 3)
kmc_13 <- perform_kmc(ps_data = ps_new,
                      ref_treatment = "PS1",
                      t_prime = "PS3",
                      K= K)
valid_data_ps_13 <- cbind(valid_data_ps, Cluster = kmc_13$clusters)
matches_1_3 <- run_matching_func(matching_method,
                                 valid_data_ps_13, ref_t=1, t_prime=3)
final_cohort_1_3 <- data.frame(id = c(rbind(matches_1_3$id_ref,
                                            matches_1_3$id_t_prime)),
                               t = c(rbind(matches_1_3$t_ref,
                                           matches_1_3$t_prime )),
                               pair = rep(seq_len(nrow(matches_1_3)),
                                          each = 2))
final_cohort_1_3 <- dplyr::left_join(final_cohort_1_3,
                                     valid_data_ps,
                                     by = c("id", "t" = "Treatment")) %>%
  dplyr::rename(Treatment = t)
abs_sb_13 <- calc_standardized_bias(data = final_cohort_1_3,
                                    covariates = covariates_for_SB,
                                    ref_treatment = 1)
############
# Pair (2 vs 3)
kmc_23 <- perform_kmc(ps_data = ps_new,
                      ref_treatment = "PS2",
                      t_prime = "PS3",
                      K = K)
valid_data_ps_23 <- cbind(valid_data_ps, Cluster = kmc_23$clusters)
matches_2_3 <- run_matching_func(matching_method, valid_data_ps_23,
                                 ref_t=2, t_prime=3)
final_cohort_2_3 <- data.frame(id = c(rbind(matches_2_3$id_ref,
                                            matches_2_3$id_t_prime)),
                               t = c(rbind(matches_2_3$t_ref,
                                           matches_2_3$t_prime )),
                               pair = rep(seq_len(nrow(matches_2_3)),
```

```
                                          each = 2))
final_cohort_2_3 <- dplyr::left_join(final_cohort_2_3,
                                     valid_data_ps,
                                     by = c("id", "t" = "Treatment")) %>%
  dplyr::rename(Treatment = t)
abs_sb_23 <- calc_standardized_bias(data = final_cohort_2_3,
                                    covariates = covariates_for_SB,
                                    ref_treatment = 2)

## 4) Combine the standardized biases
stopifnot(all(names(abs_sb_12) == names(abs_sb_13)))
stopifnot(all(names(abs_sb_12) == names(abs_sb_23)))
covariate_names <- names(abs_sb_12)
max2sb <- numeric(length(covariate_names))
for(i in seq_along(covariate_names)) {
  p_name <- covariate_names[i]
  # max over the absolute SB for (1,2), (1,3), (2,3)
  max2sb[i] <- max(abs_sb_12[p_name],
                   abs_sb_13[p_name],
                   abs_sb_23[p_name],
                   na.rm = TRUE)
}
mean_max2sb <- mean(max2sb, na.rm = TRUE)

## 5) Compute ATE(1,2) and ATE(1,3) from the *matched data*
ate_12_list <- calc_ate_ipw(data = final_cohort_1_2,
                            t1 = 1,
                            t2 = 2,
                            y_col = "Y",
                            trt_col = "Treatment",
                            ps_cols = c("PS1","PS2","PS3"))
ate_13_list <- calc_ate_ipw(data = final_cohort_1_3,
                            t1 = 1,
                            t2 = 3,
                            y_col = "Y",
                            trt_col = "Treatment",
                            ps_cols = c("PS1","PS2","PS3"))
ate_12 <- ate_12_list$ATE_t1_t2
ate_13 <- ate_13_list$ATE_t1_t2

# 6) Compute matching rate:
matched_ids_12 <- unique(final_cohort_1_2$id)
matched_ids_13 <- unique(final_cohort_1_3$id)
matched_ids_23 <- unique(final_cohort_2_3$id)
matched_union  <- unique(c(matched_ids_12, matched_ids_13, matched_ids_23))
matching_rate  <- length(matched_union) / n  # proportion matched
```

```
  return(list(max2sb = mean_max2sb,
              ate_12 = ate_12,
              ate_13 = ate_13,
              matching_rate = matching_rate))
}
```

6.5.11 Run multiple simulations

```
run_simulations <- function(B, n = 3000, K = 3) {
  # For M1
  results_M1_max2sb  <- numeric(B)
  results_M1_ate12   <- numeric(B)
  results_M1_ate13   <- numeric(B)
  results_M1_matchrate <- numeric(B)

  # For M2
  results_M2_max2sb  <- numeric(B)
  results_M2_ate12   <- numeric(B)
  results_M2_ate13   <- numeric(B)
  results_M2_matchrate <- numeric(B)

  # For VM_MD
  results_VM_MD_max2sb  <- numeric(B)
  results_VM_MD_ate12   <- numeric(B)
  results_VM_MD_ate13   <- numeric(B)
  results_VM_MD_matchrate <- numeric(B)

  for (b in seq_len(B)) {
    seed_b <- 123 + b

    # --- M1 ---
    sim_M1 <- one_simulation(
      n = n, K = K,
      matching_method  = "M1",
      seed             = seed_b
    )
    results_M1_max2sb[b]   <- sim_M1$max2sb
    results_M1_ate12[b]    <- sim_M1$ate_12
    results_M1_ate13[b]    <- sim_M1$ate_13
    results_M1_matchrate[b]<- sim_M1$matching_rate

    # --- M2 ---
    sim_M2 <- one_simulation(
      n = n, K = K,
      matching_method  = "M2",
      seed             = seed_b
    )
```

```
    results_M2_max2sb[b]   <- sim_M2$max2sb
    results_M2_ate12[b]    <- sim_M2$ate_12
    results_M2_ate13[b]    <- sim_M2$ate_13
    results_M2_matchrate[b]<- sim_M2$matching_rate

    # --- VM_MD ---
    sim_VM <- one_simulation(
      n = n, K = K,
      matching_method  = "VM_MD",
      seed             = seed_b
    )
    results_VM_MD_max2sb[b]   <- sim_VM$max2sb
    results_VM_MD_ate12[b]    <- sim_VM$ate_12
    results_VM_MD_ate13[b]    <- sim_VM$ate_13
    results_VM_MD_matchrate[b]<- sim_VM$matching_rate

    if (b %% 20 == 0) {
      cat("Finished iteration:", b, "of", B, "\n")
    }
  }
  return(list(VM = list(max2sb = results_M1_max2sb,
                        ate_12 = results_M1_ate12,
                        ate_13 = results_M1_ate13,
                        match_rate = results_M1_matchrate),
              VM2 = list(max2sb = results_M2_max2sb,
                         ate_12 = results_M2_ate12,
                         ate_13 = results_M2_ate13,
                         match_rate = results_M2_matchrate),
              VM_MD = list(max2sb = results_VM_MD_max2sb,
                           ate_12 = results_VM_MD_ate12,
                           ate_13 = results_VM_MD_ate13,
                           match_rate = results_VM_MD_matchrate)))}
```

```
# Example usage
# system.time({
#   sim_results <- run_simulations(B = 300, n=3000, K=3)
# })
# save(sim_results, file = "sim300_4var_s1_results.Rdata")
```

6.5.12 Simulation data

Since the codes running time under one scenario is roughly five minutes under 300 iterations, it would be costly to run the simulation every time. To store the simulation data that I used in this paper locally, please visit my GitHub (here), and run the five R files separately to get the corresponding data.

References

Feng, P., Zhou, X.-H., Zou, Q.-M., Fan, M.-Y., & Li, X.-S. (2012). Generalized propensity score for estimating the average treatment effect of multiple treatments. *Statistics in Medicine*, *31*(7), 681–697. https://doi.org/https://doi.org/10.1002/sim.4168

Lechner, M. (2001). Identification and estimation of causal effects of multiple treatments under the conditional independence assumption. In M. Lechner & F. Pfeiffer (Eds.), *Econometric evaluation of labour market policies* (pp. 43–58). Physica. https://www.iza.org/publications/dp/91/identification-and-estimation-of-causal-effects-of-multiple-treatments-under-the-conditional-independence-assumption

Lee, B. K., Lessler, J., & Stuart, E. A. (2010). Improving propensity score weighting using machine learning. *Statistics in Medicine*, *29*(3), 337–346. https://doi.org/10.1002/sim.3782

Lee, B. K., Lessler, J., & Stuart, E. A. (2011). Weight trimming and propensity score weighting. *PLOS ONE*, *6*, 1–6. https://doi.org/10.1371/journal.pone.0018174

Li, F., & Li, F. (2019). *Propensity score weighting for causal inference with multiple treatments*. https://arxiv.org/abs/1808.05339

Lin, L., Zhu, Y., & Chen, L. (2019). Causal inference for multi-level treatments with machine-learned propensity scores. *Health Services and Outcomes Research Methodology*, *19*(2), 106–126. https://doi.org/10.1007/s10742-018-0187-2

Lopez, M. J., & Gutman, R. (2017). Estimation of causal effects with multiple treatments: A review and new ideas. *Statistical Science*, *32*(3), 432–454. https://doi.org/10.1214/17-STS612

McCaffrey, D. F., Griffin, B. A., Almirall, D., Slaughter, M. E., Ramchand, R., & Burgette, L. F. (2013). A tutorial on propensity score estimation for multiple treatments using generalized boosted models. *Statistics in Medicine*, *32*(19), 3388–3414. https://doi.org/10.1002/sim.5753

Rubin, D. B., & Thomas, N. (1996). Matching using estimated propensity scores: Relating theory to practice. *Biometrics*, *52*(1), 249–264. https://doi.org/10.2307/2533040

Scotina, A. D., & Gutman, R. (2019). Matching algorithms for causal inference with multiple treatments. *Statistics in Medicine*, *38*(17), 3139–3167. https://doi.org/https://doi.org/10.1002/sim.8147

Zhu, Y., Coffman, D. L., & Ghosh, D. (2015). A boosting algorithm for estimating generalized propensity scores with continuous treatments. *Journal of Causal Inference*, *3*(1), 25–40. https://doi.org/10.1515/jci-2014-0022