eAppendix for the article

The right tool for the job: choosing between covariate balancing and generalized boosted model propensity scores

Keywords: Causal inference, observational study, propensity score, variable selection, diagnostic tools.

1 Summary

This eAppendix provides further details on the simulation setup, the measure of model complexity used, the standard deviation plots not reported in the paper and the comparison of the GBM and CBPS methods to the Entropy Balancing (EB) method developed by[1] for which space limitations precluded in-depth discussion in the main text. First, we reported the detailed treatment generating schemes for the seven treatment model versions spanning simple linear additive versions all the way to complex non-linear non-additive forms. We then reported the outcome generating model again spanning simple linear model up to very non-linear complex models. These detailed simulations were designed to cover a wide range of settings that we believe are often encountered by analysis. We also described our measure of model complexity with specific details on how they are estimation within each simulation scenario and sub-scenario. Lastly we reported the diagnostic plots for the standard error comparison between GBM and CBPS as well as additional comparisons of GBM and CBPS to EB. In section 5, we reported the simulation codes used in this study.

2 Simulation setup

The simulations structures used in this study are similar to the ones reported by[2] and replicated by others[3,4]. For each simulation, we generated 15 $X_1, X_2, \ldots, X_{15}$ covariates as a mixture of continuous and binary variables as described in Table 1. The continuous predictors were standard normal and the binary variables were dichotomized (at cut point 0) versions of standard normal random variables.

Table 1: Variables relationship schema and propensity estimation strategies

Variables used			Generating Models		Propensity estimation strategies					
Predictor	Dichotomized	Correlated	T	Y	1	2	3	4	5	6
X_1	x		x	x	x	x	x	x	x	x
X_2			x	x	x	x	x	x	x	x
X_3	x		x	x	x	x	x	x	x	x
X_4			x	x	x	x	x	x	x	x
X_5	x	X_1, 0.2	x			x		x	x	x
X_6	x	X_2, 0.9	x			x		x	x	x
X_7	x		x			x			x	x
X_8	x	X_3, 0.2		x			x	x	x	x
X_9		X_4, 0.9		x			x	x	x	x
X_{10}				x			x		x	x
X_{11}										x
X_{12}	x									x
X_{13}	x	X_{11}, 0.2								x
X_{14}	x	X_{12}, 0.9								x
X_{15}	x									x

Note: Correlated column includes variable correlated to and the correlation between variables. Covariates include: "direct confounders" (X_1, X_2, X_3, X_4); "distal confounders"(X_5, X_6); an "instrument" (X_7); "outcome only predictors" (X_8, X_9, X_{10}) with X_8 and X_9 distally related to the treatment; and "distractors" ($X_{11}, \ldots, X_{15}$). The Generating Models column refers to models in Tables 2 and 3 in the paper.

Simulation of treatment

We generated treatment assignments using one of seven version for the propensity score model. All seven version were of the form $Prob(T = 1) = 1/\{1 + \exp(-version - \tau\xi)\}$, where $version$ was a function of the confounders that established the complexity of the association between them and treatment assignment, ξ is a standard normal, omitted variable, uncorrelated with the covariates and outcomes, and τ the variability controlling the effect of this variable. The seven versions A through G are:

A. Additive and linear (main effects terms only):

$$Version_A = \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3 + \alpha_4 X_4 + \alpha_5 X_5 + \alpha_6 X_6 + \alpha_7 X_7$$

B. Mild non-linearity (1 quadratic term):

$$Version_B = Version_A + \alpha_8 X_2^2$$

C. Moderate non-linearity (3 quadratic terms):

$$Version_C = Version_A + \alpha_9 X_2^2 + \alpha_{10} X_4^2 + \alpha_{11} X_7^2$$

D. Mild non-additivity (4 two-way interaction terms):

$$Version_D = Version_A + \alpha_{12} X_1 X_3 + \alpha_{13} X_2 X_4 + \alpha_{14} X_4 X_5 + \alpha_{15} X_5 X_6$$

E. Mild non-additivity and non-linearity (1 quadratic term and 4 two-way interaction terms):

$$Version_E = Version_D + \alpha_{16} X_2^2$$

F. Moderate non-additivity (9 two-way interaction terms):

$$Version_F = Version_D + \alpha_{17} X_5 X_7 + \alpha_{18} X_1 X_6 + \alpha_{19} X_2 X_3 + \alpha_{20} X_3 X_4 + \alpha_{21} X_4 X_5$$

G. Moderate non-linearity and non-additivity (3 quadratic term and 9 two-way interaction terms):

$$Version_G = Version_F + \alpha_{22} X_2^2 + \alpha_{23} X_4^2 + \alpha_{24} X_7^2$$

The parameters $\alpha_1, \alpha_2, \ldots, \alpha_{24}$ have values between -0.8 and 0.8. For each version, 6 different values of τ were considered ($\tau = 0, 0.25, 0.5, 1, 1.5, 2$) to allow for variability in the predictive power of the propensity score model of the observed covariates.

Simulation of outcome

Next, five different outcomes relationships were considered with the association between outcome and covariates ranging from linear additive to non-linear as followed:

Outcome 1. Additive and linear (main effects terms only):

$$Y = \delta_0 + \theta T + \delta_1 X_1 + \delta_2 X_2 + \delta_3 X_3 + \delta_4 X_4 + \delta_5 X_8 + \delta_6 X_9 + \delta_7 X_{10} + \sigma\zeta_1$$

Outcome 2. Mild non-linearity (2 individual non-linear variables):

$$Y = \delta_0 + \theta T + \delta_1 X_1 + \delta_2 X_2^2 + \delta_3 X_3 + \delta_4 \exp(1.3 X_4) + \delta_5 X_8 + \delta_6 X_9 + \delta_7 X_{10} + \sigma\zeta_2$$

Outcome 3. Moderate non-linearity (exponential interaction among all confounders):

$$Y = \delta_0 + \theta T + \exp(\delta_1 X_1 + \delta_2 X_2 + \delta_3 X_3 + \delta_4 X_4 + \delta_5 X_8 + \delta_6 X_9 + \delta_7 X_{10}) + \sigma\zeta_3$$

Outcome 4. Severe non-linearity (non-linear, exponential interaction and additive terms):

$$Y = \delta_0 + \theta T + \exp(\delta_1 X_1 + \delta_2 X_2 + \delta_3 X_3) + \delta_4 \exp(1.3 X_4) + \delta_5 X_8 + \delta_6 X_9 + \delta_7 X_{10} + \sigma\zeta_4$$

Outcome 5. Sinusoidal non-linearity (Sine function of confounders):

$$Y = \delta_0 + \theta T + 4\sin(\delta_1 X_1 + \delta_2 X_2 + \delta_3 X_3 + \delta_4 X_4 + \delta_5 X_8 + \delta_6 X_9 + \delta_7 X_{10}) + \sigma\zeta_5$$

The treatment effect θ was assumed additive and equal to -0.4. The outcome random error was normally distributed with mean 0 and variance σ^2. For each outcome specification 20 different σ are considered ranging from 0 to 5 (0.25 point increment) to allow for greater variability in the in the unmeasured random term ξ of the outcome model. The simulation parameters $\delta_1, \delta_2, \ldots, \delta_7$ were selected with values between -0.73 and 0.71 and the intercept $\delta_0 = -3.85$. For each scenario, a sample of 1000 observations was generated and for each scenario we generated 1000 replicate data sets.

3 Details of the proposed model complexity measure

To assess the impact of the complexity of the data generating structures the relative performance of CBPS and GBM, we measure the proportion of variance that could have been explained in a model (R-square) if polynomial models were considered, relative to the linear specification of the logistic regression model. For the complexity of the relationship between the outcome and the predictors we used the R-squares from linear models and for the complexity in the propensity score model, we used pseudo R-squares from logistic regression models[5;6]. For a simulated data and a specific strategy, the following steps are used to measure complexity

- First, a model was fitted with the strategy predictors as specified in Table 1 in the paper linearly entered in the regression and the model R-square (called R_0^2) recorded. For example, when using the estimation strategy 1, the variables X_1, X_2, X_3, X_4 are included in a linear regression with Y as the outcome or in a logistic regression with T as the outcome. In the outcome model the treatment variable T is also included as predictor in the model.

- Then a second model was fitted with the linear predictors as well as their quadratic and cubic power and all interactions between the continuous predictors included. The model R-square R_1^2 was also recorded. For strategy 1, the model will include X_1, X_2, X_3, X_4, as well as X_2^2, X_4^2 (quadratic), X_2^3, X_4^3 (cubic) and the interaction X_2X_4. This model will potentially capture most non-linearity that may exist in the relationship.

- A model complexity R_d^2 was then estimated as the difference in R-square

$$R_d^2 = R_1^2 - R_0^2$$

  between the two estimates where values close to 0 will reveal data models that can be well fitted without needing higher order polynomial terms and large values will reflect very complex model that cannot be covered by simply using available covariates without the higher order terms. For the outcome and propensity score models, we will denote such complexity indices by R_{dy}^2 and R_{dt}^2 respectively and they will be specific to each data generated.

The impact of model complexity on bias, root mean square error (RMSE) and standard deviation (SD) will be contrasted between the different values of R_{dy}^2 and R_{dt}^2 but within each scenario (where bias, RMSE and standard deviation are computed) each simulated data can lead to slightly different complexity measures. To assure that an estimate of bias,

RMSE and standard deviation is associated with simulated data with very similar complexities, the 1,000 simulations within each scenario were divided into 9 sub-scenarios on the basis of the complexities R^2_{dy} and R^2_{dt}. Within a scenario, the distribution of R^2_{dy} and R^2_{dt} each were split into 3 groups each, the first between the 5^{th} and the 35^{th} percentile, the second between the 35^{th} and the 65^{th} percentile and the last one between the 65^{th} and the 95^{th} percentile. We defined the 9 sub-scenarios through cross-classification of the 3 groups for R^2_{dy} and the 3 groups for R^2_{dt}. Simulated data with outcome or propensity score complexity below the 5^{th} or above the 95^{th} percentile on either R_{dy} or R_{dt} were considered outliers and dropped from the model complexity results. For the 9 sub-scenarios formed within scenarios, the means of R^2_{dy} and R^2_{dt} were used as average complexities associated with the bias, RMSE and standard deviation computed within such sub-scenario (i.e. using only simulated data that fall within the sub-scenario). There was an average of 100 simulated data in each sub-scenario and the average standard deviation in R^2_{dy} and R^2_{dt} across all sub-scenarios was evaluated at 0.005 and 0.002 respectively, thus making the majority data level complexity in each sub-scenario within 0.01 point or less of each others. This assured that all the simulations used within each estimation of bias, RMSE or standard deviation have very similar complexities.

4 Variability in inverse-probability weights

In propensity score methods, the variance of the inverse-probability weights has the potential to increase the variability of estimated treatment effects and can indicate large influence for a small number of observations. For each simulated dataset, we estimated the sample standard deviation of the weights for the control group and report the average over the datasets in each scenario in the result tables in the paper. We also compared the variability in the weights between GBM and CBPS by estimating the ratio of the standard deviation of the weights between the methods. The summary statistics are reported in Table2

Table 2: Ratio of weight average standard deviation between CBPS/GBM

Propensity Strategy	Summary Statistics					
	Mean	Minimum	$10^{th}pctl$	Median	$90^{th}pctl$	Maximum
1	1.049	0.817	0.817	1.037	1.263	1.263
2	1.341	0.934	0.934	1.422	1.586	1.586
3	1.121	0.879	0.879	1.102	1.347	1.347
4	1.182	0.969	0.969	1.193	1.352	1.352
5	1.402	0.997	0.997	1.475	1.642	1.642
6	1.490	1.092	1.092	1.551	1.724	1.724

Note: $pctl$ = percentile

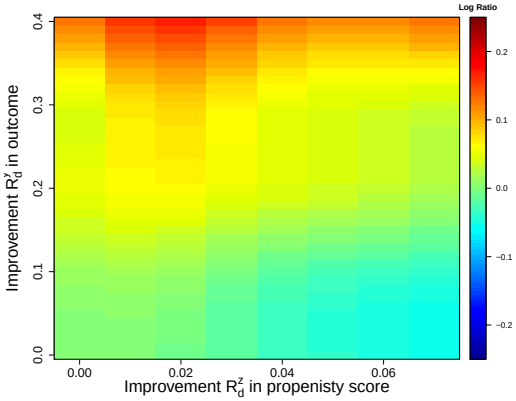
For nearly all propensity estimation strategies, CBPS yields more variable weights on average but this does not necessarily translate to larger standard errors for the treatment effect: the standard errors are more similar across CBPS and GBM than the variance of the weights and more often smaller for CBPS than GBM. Hence, variance in the weights did not necessarily translate to more variable treatment effect estimates because the improved covariate balance provided by CBPS tended to reduce the variance of the estimated treatment effects.

5 Method diagnostic standard error plots between GBM and CBPS

The comparison of bias and RMSE performance statistics as they related to the model complexity indicators were reported in the paper. In this section we reported the estimates for standard errors (Figure 1) under the model strategies 1 (predictors of both Y and T), 2 (predictors of T regardless of Y) and 6 (all predictors included) as well as the average across all strategies. In general, the standard error performances were similar between GBM and CBPS. The log of the ratios of standard errors were all between -0.25 and 0.25, so that the standard error ratios are between 0.78 and 1.28. Even then, for the large majority of scenarios, the log of ratio of standard errors has a very low value between -0.15 and 0.15 which reflect the standard error in one method mostly being up to more than 1.16 times larger or smaller than those in the other.

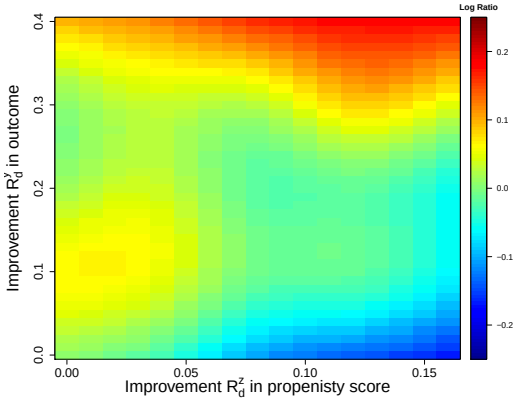
a. Std Error Strategy 1: Predictors of both Y and T

Heat density scale: -0.25, 0.25



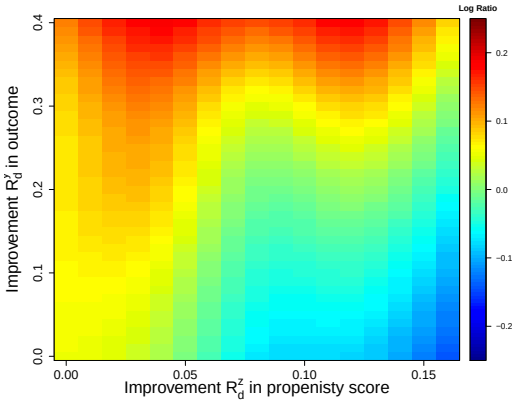
b. Std Error Strategy 2: Predictors of both Y and T

Heat density scale: -0.25, 0.25



c. Std Error for all predictors included

Heat density scale: -0.25, 0.25



d. Std Error averaged across all strategies

Heat density scale: -0.25, 0.25

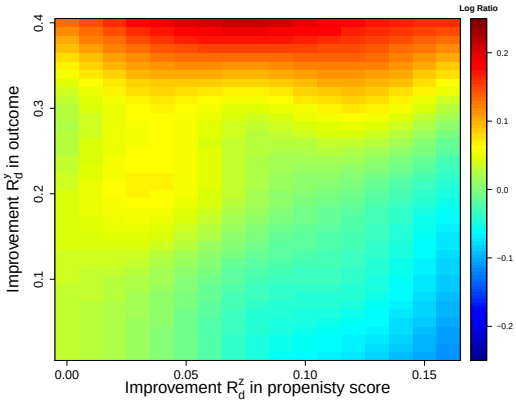


Figure 1: Simulation results on standard errors comparing GBM to CBPS under different strategies and a summary across all strategies.

6 Comparisons with the Entropy Balancing method

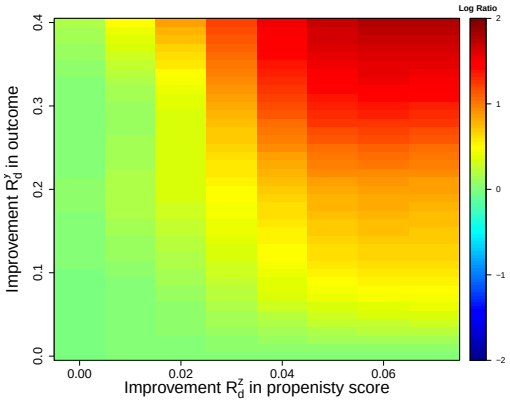
Similar to the comparison conducted comparing GBM to CBPS, we also applied the entropy balancing method to the simulated data and assessed comparisons between GBM and entrpy balancing as well as between CBPS and entropy balancing. The entropy balancing method was setup to calibrate weights to be used in analyses so that the reweighted treatment and control group satisfy a potentially large set of prespecified balance conditions that incorporate information about known sample moments. It uses on a maximum entropy reweighting theory and thereby exactly adjusts inequalities in representation with respect to the first, second, and possibly higher moments of the covariate distributions. While the inverse probability weights in the propensity score methods are calculated via CBPS or GBM, entropy balancing, in common with many traditional survey weighting schemes, directly calculates weights to adjust for known sample distributions, so integrating covariate balance directly into the weights and avoiding the model checking required in GBM and CBPS. As this method approaches the issue of balancing treatment and control groups from a different setting, we report the results of the comparison between the entropy method and the two other methods discussed in this study.

6.1 Comparisons between GBM and EB

For these results, the log-relative values of the statistic (RMSE, Bias and Standard Deviation) was computed as $log(STAT_{EB}/STAT_{GBM})$ where for the statistic of interest, $STAT_{EB}$ was the estimated value for EB and $STAT_{GBM}$ was the estimated value for GBM. Negative values of the log-relative statistics were indicative of better performance of EB when compared to GBM and positive values indicative of dominance by GBM. Similar to the comparison between GBM and CBPS, when there is confident about the confounders that explain treatment assignment and the complexity in the propensity model is low, EB will likely produce better results as it has RMSEs similar to the ones obtained when using GBM while at the same time enjoying less bias as in the case of CBPS. For example, for the average comparison across all the strategies, model complexities $R^2_{dy} = 0.01$ and $R^2_{dt} = 0.01$ resulted on an average log of relative RMSE of 0.03 suggesting that the EB method has an RMSE 1.03 time smaller than the GBM RMSE while at the same time it resulted in a relative bias of -1.43 which is equivalent to the EB method having a bias 4.17 times smaller. Conversely, when complexity in the propensity score model and/or outcome model is high, GBM will outperform EB (Appendix Figure 2). For $R^2_{dy} = 0.1$ and $R^2_{dt} = 0.1$, log of relative RMSE, bias and standard deviation were 0.39, 4.14 and 0.002 respectively equivalent to GBM producing RMSE, bias and standard deviation 1.48, 62.59 and 1.002 times higher respectively. In most cases, the standard deviations were similar between the methods with extreme results only occurring when the outcome model is extremely complex and GBM slightly outperforms EB (in the order of 1.28 times).

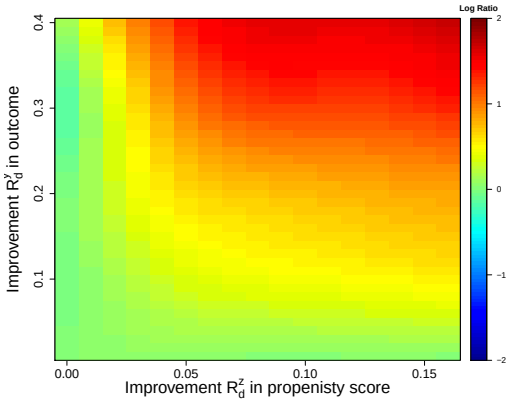
a. RMSE Strategy 1: Predictors of both Y and T

Heat density scale: -2, 2



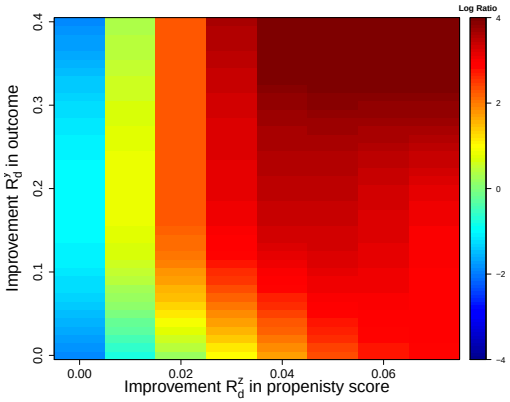
d. RMSE averaged across all strategies

Heat density scale: -2, 2



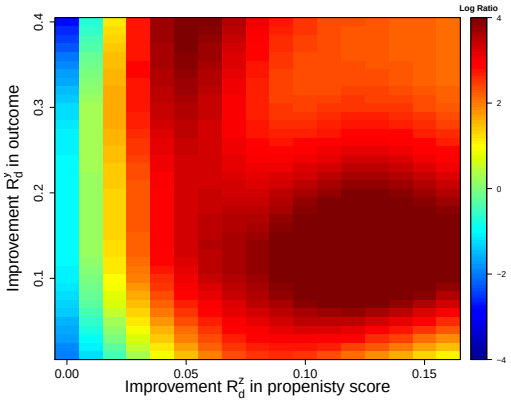
b. Bias Strategy 1: Predictors of both Y and T

Heat density scale: -4, 4



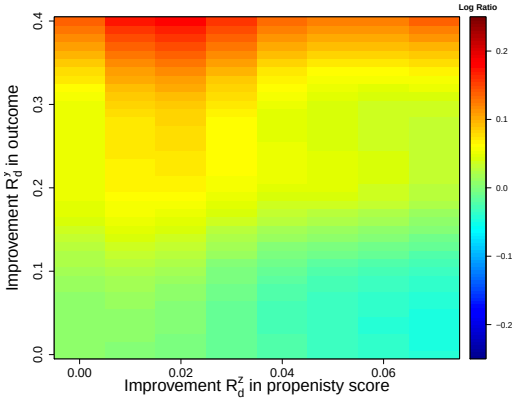
e. Bias averaged across all strategies

Heat density scale: -4, 4



c. Std Strategy 1: Predictors of both Y and T

Heat density scale: -0.25, 0.25



f. Std averaged across all strategies

Heat density scale: -0.25, 0.25

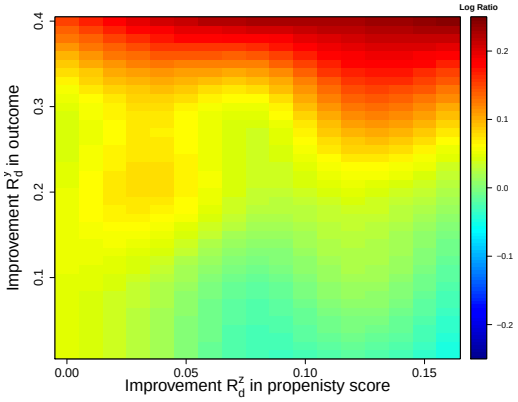


Figure 2: Simulation results comparing GBM to Entropy balancing under strategy 1 (including predictors of treatment T and outcome Y) and a summary across all strategies..

6.2 Comparisons between CBPS and EB

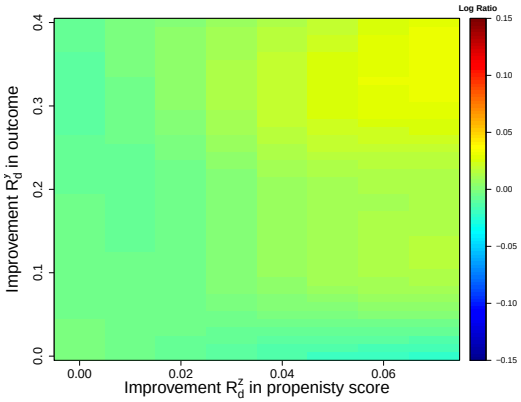
For comparing CBPS and EB, the log-relative statistic was defined as ($log(STAT_{CBPS}/STAT_{EB})$) where negative values will reflect better performance of EB and positive values will be indicative of CBPS performing better. The result of the comparison is reported in (Appendix Figure 3). Overall, with the exception of few areas, EB and CBPS performed similarly. The log-relative statistics were mostly between -0.15 and 0.15 for both RMSE and standard deviation (equivalent to one method performing better than the other for up to 16% more) and between -0.5 and 0.5 for the bias (or a better performance up to more than 65%). The extremes are particularly rare. On the average statistics across all strategies, the extreme RMSE comparison observed was for $R^2_{dy} = 0.05$ and $R^2_{dt} = 0.15$ with a log-relative RMSE of -0.049 equivalent to CBPS producing a RMSE 5% smaller than EB. At $R^2_{dy} = 0.30$ and $R^2_{dt} = 0.07$ the estimate RMSE from EB was 3% smaller than the one from CBPS (log-relative RMSE 0.029). These small differences were also observed in the standard deviation where the extreme difference observed at $R^2_{dy} = 0.10$ and $R^2_{dt} = 0.15$ shows CBPS producing a standard deviation 6.3% smaller (log-relative standard deviation -0.06). For the bias, some moderate differences are observed in some spots (e.g. $R^2_{dy} = 0.03$ and $R^2_{dt} = 0.14$ or $R^2_{dy} = 0.40$ and $R^2_{dt} = 0.00$) but those seemed to be exceptions. In most cases, EB slightly outperformed CBPS in bias.

6.3 Summary of performances between GBM, CBPS and EB

Overall, the results of the comparison of the three methods suggest that the performance of CBPS and EB are very similar across most simulations. But as reported in the paper, when an analyst is confident about the confounders or variables that determine treatment assignment and the complexity in the propensity model is low, CBPS and EB are similar and will likely produce better results as they have RMSEs similar to the ones obtained when using GBM while at the same time enjoying less bias. Conversely, GBM will perform better then both methods when complexity in the propensity score model and/or outcome model is high.

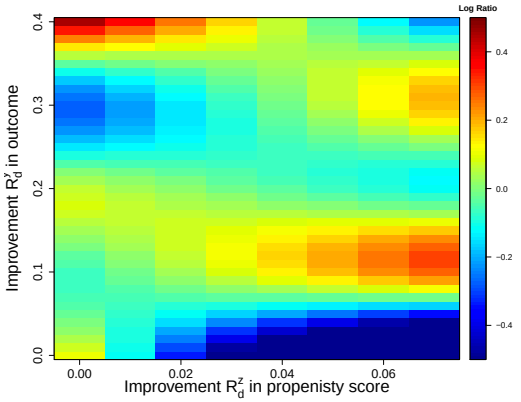
a. RMSE Strategy 1: Predictors of both Y and T

Heat density scale: -0.15, 0.15



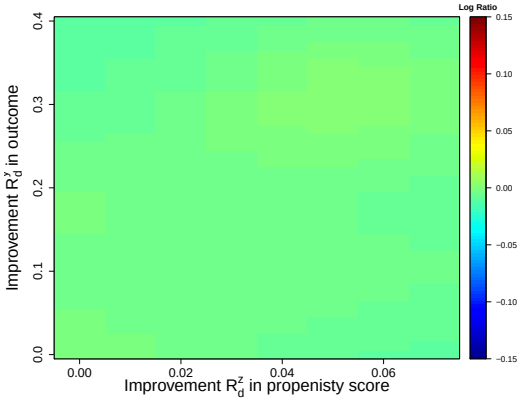
b. Bias Strategy 1: Predictors of both Y and T

Heat density scale: -0.5, 0.5



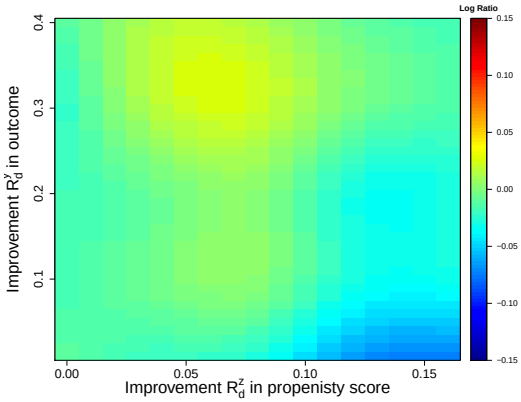
c. Std Strategy 1: Predictors of both Y and T

Heat density scale: -0.15, 0.15



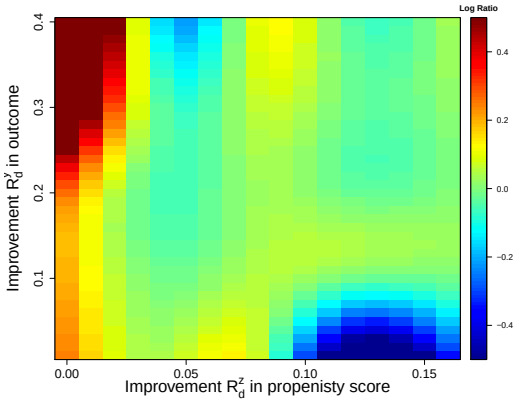
d. RMSE averaged across all strategies

Heat density scale: -0.15, 0.15



e. Bias averaged across all strategies

Heat density scale: -0.5, 0.5



f. Std averaged across all strategies

Heat density scale: -0.15, 0.15

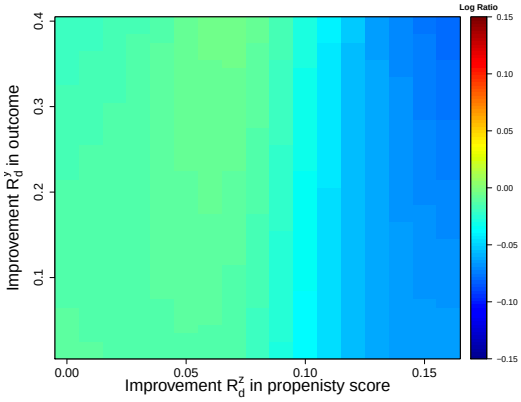


Figure 3: Simulation results comparing CBPS to Entropy balancing under strategy 1 (including predictors of treatment T and outcome Y) and a summary across all strategies..

7 Simulation codes

Below, we reported the different simulation codes used for the simulations reported in this study. There are 4 codes in R and one code in SAS. The first R code collects all the runs to be done by calling different functions. The second one is the collection of all the data generating functions, the third has the estimation functions and the last one is the collection of summary functions in R. For reasons of run times, after the propensity score weights are computed from each method, the SAS code is used to generate additional outcomes for each simulated data and estimates computed.

7.1 Runs codes in the file "SimulationRun.R"

```
# SUPPORTING INFORMATION TEXT for Setodji et al, 2017
#~~~~~~~~~~~~~~~~~~~~Overview~~~~~~~~~~~~~~~~~~~~#
# Author: Claude Setodji
# This script is used for running the different functions
# for the simulation
#   1- Simulate data
#   2- Get propensity score estimates from different methods
#   3- Get summary statistics from the estimates
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#



### Set the directories for outputing simulation esitmates
claude <-c("140121")
proj.dir <- "/vincent/b/setodji/causal/"
dat.dir <- paste(proj.dir,"Data/",sep="")
code.dir <- paste(proj.dir,"Code/",sep="")
out.dir <- paste(proj.dir,"Output/",sep="")

## Set working directory
setwd(proj.dir)
## Set number of simulations and number of observations in each dataset
   to be simulated
 nsim <- 1000
 nobs <- 1000

covariatelist <- list(
  paste("w",c(1:4),sep=""),
  paste("w",c(1:7),sep=""),
  paste("w",c(1:4,8:10),sep=""),
  paste("w",c(1:6,8:9),sep=""),
  paste("w",c(1:10),sep=""),
  paste("w",c(1:15),sep="")
)


#schema list
scenarios <- c("a","b","c","d","e","f","g")
#versions <- 1:6
versions <- 1:length(covariatelist)
```

```
cmethods <- c("lrg","gbm","cbps")
#cmethods <- c("lrg")

## Generate the estimates under different sigma scenarios
## This part of the process takes a long time and so these loops should
   be run preferably separately.
for(irr in 1:6){
## Set the output name for each run with different sigma
## The mysigma adds error variable to the estimation of the propensity
   score
 claude <- paste("LeeS", irr, "Last", nobs, sep="")
 if(irr == 1){ set.seed(1508071)
               mysigma <- 0
     }
 if(irr == 2){ set.seed(1508072)
                mysigma <- 0.25
     }
 if(irr == 3){ set.seed(1508073)
                mysigma <- 0.5
     }
 if(irr == 4){ set.seed(1508124)
                mysigma <- 1
     }
 if(irr == 5){ set.seed(1508125)
                mysigma <- 1.5
     }
 if(irr == 6){ set.seed(1508126))
                mysigma <- 2
     }

## Now run the
## 1- Simulate data: This only simulate linear outcomes.
##    Additional simulated outcomes will be done using the saved
##    datasets after the propopensity score weights are estiamted
##    in order to avoid the run time to be impossible to control.
##    These additional outcome runs will be done in SAS for
##    processing speed.

 source("GenerateData.R")

## 2-Compute the propensity score weight under different methods

 source("FEstimation.R")

## Compute summary statistics such as ASMD, KS, Bias, MSE
 source("FSummary.R")


}
```

7.2 Data generating functions in the file “GenerateData.R”

```
# SUPPORTING INFORMATION TEXT for Setodji et al, 2017
#~~~~~~~~~~~~~~~~~~~~Overview~~~~~~~~~~~~~~~~~~~~#
# Author: Claude Setodji
# This script is used for dataset generation for
# the paper "The right tool for the job: choosing
#   between covariate balancing and generalized boosted
#   model propensity scores" in Epidemiology 2017.
# The study design is based on methods printed in
# Setoguchi et al., "Evaluating uses of data mining
# techniques in propensity score estimation: a
# simulation study." Pharmacoepi Drug Saf 2008.

#~~~~~~~~~~~~~~~~~~~~Functions~~~~~~~~~~~~~~~~~~~~#
# function: generate continuous random variable correlated to variable
# x by rho invoked by the "F.generate" function
# Parameters -
#    x - data vector
#    rho - correlation coefficient
# Returns -
#    a correlated data vector of the same length as x

    F.sample.cor <- function(x, rho) {
            y <- (rho * (x - mean(x)))/sqrt(var(x)) + sqrt(1 - rho^2) *
                rnorm(length(x))
            return(y)
    }

# function: generate simulation datasets
# inputs: sample size N, scenario
# outputs: 1 dataset of size N
    # binary variables: w1, w3, w5, w6, w8, w9
    # continous variables: w2, w4, w7, w10
    # confounders: w1, w2, w3, w4
    # exposure predictors only: w5, w6, w7
    # outcome predictors only: w8, w9, w10
    # correlations: (w1,w5)=0.2, (w2,w6)=0.9, (w3,w8)=0.2, (w4,w9)=0.9

    F.generate <- function(size, scenario) {
        w1 <- rnorm(size, mean=0, sd=1)
        w2 <- rnorm(size, mean=0, sd=1)
        w3 <- rnorm(size, mean=0, sd=1)
        w4 <- rnorm(size, mean=0, sd=1)
        w5 <- F.sample.cor(w1, 0.2)
        w6 <- F.sample.cor(w2, 0.9)
        w7 <- rnorm(size, mean=0, sd=1)
        w8 <- F.sample.cor(w3, 0.2)
        w9 <- F.sample.cor(w4, 0.9)
        w10 <- rnorm(size, mean=0, sd=1)


  #w11-w25 independent of everything (distractors)
```

```
w11 <- rnorm(size, mean=0, sd=1)
w12 <- rnorm(size, mean=0, sd=1)
w13 <- F.sample.cor(w11, 0.2)
w14 <- F.sample.cor(w12, 0.9)
w15 <- rnorm(size, mean=0, sd=1)


   #~~ dichotomize variables (will attenuate correlations above)
      w1 <- ifelse(w1 > mean(w1), 1, 0)
      w3 <- ifelse(w3 > mean(w3), 1, 0)
      w5 <- ifelse(w5 > mean(w5), 1, 0)
      w6 <- ifelse(w6 > mean(w6), 1, 0)
      w8 <- ifelse(w8 > mean(w8), 1, 0)
      w9 <- ifelse(w9 > mean(w9), 1, 0)
      w13 <- ifelse(w13 < mean(w13),1,0)
      w14 <- ifelse(w14 > mean(w14), 1, 0)
      w15 <- ifelse(w15 > mean(w15), 1, 0)

   #~~ scenarios for data generation models
      # A: model with additivity and linearity
      # B: mild non-linearity
      # C: moderate non-linearity
      # D: mild non-additivity
      # E: mild non-additivity and non-linearity
      # F: moderate non-additivity
      # G: moderate non-additivity and non-linearity

  # binary exposure modeling
    #scenario A
          z.a_trueps <- b1*w1 + b2*w2 + b3*w3 + b4*w4 + b5*w5 + b6*w6
             + b7*w7 + mysigma*rnorm(size, mean=0, sd=1)
          z.a_trueps <- (1 + exp( -(0 + z.a_trueps) ) )^-1
    #scenario B
          z.b_trueps <- b1*w1 + b2*w2 + b3*w3 + b4*w4 + b5*w5 + b6*w6
             + b7*w7 + b2*w2*w2 + mysigma*rnorm(size, mean=0, sd=1)
          z.b_trueps <- (1 + exp( -(0 + z.b_trueps) ) )^-1
    #scenario C
          z.c_trueps <- b1*w1 + b2*w2 + b3*w3 + b4*w4 + b5*w5 + b6*w6
             + b7*w7 + b2*w2*w2 +b4*w4*w4 + b7*w7*w7 +
             mysigma*rnorm(size, mean=0, sd=1)
          z.c_trueps <- (1 + exp( -(0 + z.c_trueps) ) )^-1
    #scenario D
          z.d_trueps <- b1*w1 + b2*w2 + b3*w3 + b4*w4 + b5*w5 + b6*w6
             + b7*w7 + b1*0.5*w1*w3 + b2*0.7*w2*w4 + b4*0.5*w4*w5 +
             b5*0.5*w5*w6 + mysigma*rnorm(size, mean=0, sd=1)
          z.d_trueps <- (1 + exp( -(0 + z.d_trueps) ) )^-1
    #scenario E
          z.e_trueps <- b1*w1 + b2*w2 + b3*w3 + b4*w4 + b5*w5 + b6*w6
             + b7*w7 + b2*w2*w2 + b1*0.5*w1*w3 + b2*0.7*w2*w4 +
             b4*0.5*w4*w5 + b5*0.5*w5*w6 + mysigma*rnorm(size,
             mean=0, sd=1)
          z.e_trueps <- (1 + exp( -(0 + z.e_trueps) ) )^-1
    #scenario F
```

```
      z.f_trueps <- b1*w1 + b2*w2 + b3*w3 + b4*w4 + b5*w5 + b6*w6
         + b7*w7 + b1*0.5*w1*w3 + b2*0.7*w2*w4 + b3*0.5*w3*w5 +
         b4*0.7*w4*w6 + b5*0.5*w5*w7 + b1*0.5*w1*w6 +
         b2*0.7*w2*w3 + b3*0.5*w3*w4 + b4*0.5*w4*w5 +
         b5*0.5*w5*w6 + mysigma*rnorm(size, mean=0, sd=1)
      z.f_trueps <- (1 + exp( -(0 + z.f_trueps) ) )^-1
#scenario G
      z.g_trueps <- b1*w1 + b2*w2 + b3*w3 + b4*w4 + b5*w5 + b6*w6
         + b7*w7 + b2*w2*w2 + b4*w4*w4 + b7*w7*w7 + b1*0.5*w1*w3
         + b2*0.7*w2*w4 +b3*0.5*w3*w5 + b4*0.7*w4*w6 +
         b5*0.5*w5*w7 + b1*0.5*w1*w6 + b2*0.7*w2*w3 +
         b3*0.5*w3*w4 + b4*0.5*w4*w5 + b5*0.5*w5*w6 +
         mysigma*rnorm(size, mean=0, sd=1)
      z.g_trueps <- (1 + exp( -(0 + z.g_trueps) ) )^-1
#scenario H
      z.h_trueps <- b1*w11 + b2*w12 + b3*w13 + b4*w14 + b5*w15 +
         b6*w6 + b7*w7 + b2*w12*w12 + b4*w14*w14 + b7*w7*w7 +
         b1*0.5*w11*w13 + b2*0.7*w12*w14 +b3*0.5*w13*w15 +
         b4*0.7*w14*w6 + b5*0.5*w5*w7 + b1*0.5*w11*w6 +
         b2*0.7*w12*w13 + b3*0.5*w13*w14 + b4*0.5*w14*w15 +
         b5*0.5*w15*w6 + mysigma*rnorm(size, mean=0, sd=1)
      z.h_trueps <- (1 + exp( -(0 + z.h_trueps) ) )^-1
#scenario I
      z.i_trueps <- b1*w11 + b2*w12 + b5*w15 + b6*w6 + b7*w7 +
         b2*w12*w12 + b1*0.5*w11*w12 + b2*0.7*w12*w15 +
         b1*0.5*w11*w6 + b2*0.7*w12*w13 + b4*0.5*w14*w15 +
         b5*0.5*w15*w6 + mysigma*rnorm(size, mean=0, sd=1)
      z.i_trueps <- (1 + exp( -(0 + z.i_trueps) ) )^-1
#scenario J
      z.j_trueps <- b1*w1 + b1*w11 + b2*w12 + b5*w15 + b6*w6 +
         b2*w12*w12 + b1*0.5*w11*w12 + b2*0.7*w12*w15 +
         b1*0.5*w11*w1 + b2*0.7*w12*w13 + b4*0.5*w14*w15 +
         b5*0.5*w15*w1 + mysigma*rnorm(size, mean=0, sd=1)
      z.j_trueps <- (1 + exp( -(0 + z.j_trueps) ) )^-1

  # probability of exposure: random number betw 0 and 1
  # if estimated true ps > prob.exposure, than received exposure
     (z.a=1)
      prob.exposure <- runif(size)
      z.a <- ifelse(z.a_trueps > prob.exposure, 1, 0)
      z.b <- ifelse(z.b_trueps > prob.exposure, 1, 0)
      z.c <- ifelse(z.c_trueps > prob.exposure, 1, 0)
      z.d <- ifelse(z.d_trueps > prob.exposure, 1, 0)
      z.e <- ifelse(z.e_trueps > prob.exposure, 1, 0)
      z.f <- ifelse(z.f_trueps > prob.exposure, 1, 0)
      z.g <- ifelse(z.g_trueps > prob.exposure, 1, 0)
      z.h <- ifelse(z.h_trueps > prob.exposure, 1, 0)
      z.i <- ifelse(z.i_trueps > prob.exposure, 1, 0)
      z.j <- ifelse(z.j_trueps > prob.exposure, 1, 0)

  # continuous outcome modeling
      y.a <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
         a7*w10 + g1*z.a
```

```
            y.b <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.b
            y.c <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.c
            y.d <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.d
            y.e <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.e
            y.f <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.f
            y.g <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.g
            y.h <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.h
            y.i <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.i
            y.j <- a0 + a1*w1 + a2*w2 + a3*w3 + a4*w4 +a5*w8 + a6*w9 +
                a7*w10 + g1*z.j

        # create simulation dataset
            sim <- as.data.frame(cbind(w1, w2, w3 ,w4, w5, w6, w7, w8,
                w9, w10, w11, w12, w13, w14, w15,
                                z.a, z.b,z.c,z.d,z.e,z.f,z.g,
                                y.a,y.b,y.c,y.d,y.e,y.f,y.g))
            return(sim)
    }

#~~~~~~~~~~~~~~~Global Variables~~~~~~~~~~~~~~~~~~~#

    #~~ coefficients for data generation models
        b0 <- 0
        b1 <- 0.8
        b2 <- -0.25
        b3 <- 0.6
        b4 <- -0.4
        b5 <- -0.8
        b6 <- -0.5
        b7 <- 0.7
        a0 <- -3.85
        a1 <- 0.3
        a2 <- -0.36
        a3 <- -0.73
        a4 <- -0.2
        a5 <- 0.71
        a6 <- -0.19
        a7 <- 0.26
        g1 <- -0.4 # effect of exposure

#~~~~~~~~~~~~~~~~~~~~Calls~~~~~~~~~~~~~~~~~~~~#

    # this generates datasets
    # Example: Generate 1000 datasets of N=500 in scenario G
 #    simdata <- replicate(1000, F.generate(500, "G"))
 #   simdata <- replicate(2, F.generate(500, "G"))
```

```
simdata <- lapply(1:nsim, function(i) {F.generate(nobs)})
```

7.3 Estimation functions in the file "FEstimation.R"

```
#####################################################
#Functions for estimation File
#####################################################

#Author: Claude Setodji
#setodji@rand.org

# Adaptation of Brian Lee's code used for the simulation on their paper
# "Weight trimming and propensity score weighting" in PLoS ONE 2011.
# Most of the functions were rewritten to fit the simulations conducted
    here

###################
### Packages needed
library(twang)
library(rpart)
library(ipred)
library(randomForest)
library(CBPS)
###################

# function: GBM estimation of propensity score / this is optimized for
    ATT estimation
F.gbm.sim <- function(dataset,estimand,assign,version) {
  dataset$z <- dataset[,assign]
  start.time <- proc.time()[3]
  mmvar <-c("z", covariatelist[[version]])
  dataseto <- dataset[,mmvar]
  formula <- F.zformula(version, covariatelist=covariatelist)
  ps.model <- ps(formula,
                 data=dataseto, title="none", stop.method="ks.mean",
                 n.trees=20000, interaction.depth=4,
                 shrinkage=0.0005, verbose=FALSE,estimand=estimand,
                 cv.folds=10)

  # CPU processing time
  elapsed <- proc.time()[3]-start.time
  cat("GBM Elapsed: ", elapsed, "s  ")

  # returns gbm model
  return(ps.model)
}

# function: return the pscore model formula for a particular version

chartoform <- function(respname,xname){
   as.formula(paste(respname," ~ ",paste(xname,collapse=" + ")))
 }

# This will allow us to setup as many versions as we want at the end
#  using covariatelist
```

```
F.zformula <- function(version, covariatelist=covariatelist) {
  returnval <- NULL
  returnval <- chartoform("z",covariatelist[[version]])
return(returnval)
}


# function: logistic regression estimation of propensity score

F.lrg.sim <- function(dataset,assign,version) {
  dataset$z <- dataset[,assign]
  formula <- F.zformula(version, covariatelist=covariatelist)
  ps.model <- glm(formula, data=dataset, family=binomial(link="logit"))
  phat <- predict(ps.model)
  phat[phat == 1] <- 0.999
  phat[phat == 0] <- 0.001
  odds <- exp(predict(ps.model))
  return.obj <- list(ps.model=ps.model,pscore=odds/(1+odds))
  return(return.obj)
}


# function: CBPS logistic regression estimation of propensity score
    using Imai and Ratkovic's method

F.cbps.sim <- function(dataset,estimand,assign,version) {
  dataset$z <- dataset[,assign]
  mmvar <-c("z", covariatelist[[version]])
  dataseto <- dataset[,mmvar]
  start.time <- proc.time()[3]
  formula <- F.zformula(version, covariatelist=covariatelist)
  ps.model <- glm(formula, data=dataseto, family=binomial(link="logit"))
  amt <- TRUE
  if(estimand == "ATE"){amt <- FALSE}
  ps.model <- CBPS(formula, data=dataset, ATT = amt)
  ps.model$fitted.values[ps.model$fitted.values == 1] <- 0.999
  ps.model$fitted.values[ps.model$fitted.values == 0] <- 0.001

  # CPU processing time
  elapsed <- proc.time()[3]-start.time
  cat("CBPS Elapsed: ", elapsed, "s  ")

  # returns gbm model
  return(ps.model)
}

## function prep for the Heinmueller method of Entropy Balance (EB)

mdia <- function(x, target, sweights=rep(1,nrow(x)), maxdelta=1,
                 maxiter=1000, tol=1e-08){
  ###################################################################
  ## x - a dataframe or matrix
  ## target - a vector of length(nool(x))
```

```
## sweights -- optional sampling weights,
## maxdelta -- maximum step size for Newton-Raphson
## maxiter -- max iteration
## tol -- tolerance for convergence and checking that solution yields
##  weights that achieve balance
## Value:
## beta -- coefficients of MDIA weight function
## se_beta -- Standard error of beta
## covbeta -- variance-covariance matrix for beta
## weights -- MDIA weights
## exitcode -- 0: solution found and it yields balance;
##             -1: no solution, probably not in convex hull;
##             -2: solution that didn't yield balance
##############################################################
stopifnot(is.numeric(target), ncol(x)==length(target),
   nrow(x)==length(sweights), maxiter > 0, maxdelta > 0, tol>0)
.mdia <- function(x, target, sweights=rep(1,nrow(x)), maxdelta=1,
   maxiter=1000, tol=1e-08){
    x <- as.matrix(x)
    n <- nrow(x)
    m <- ncol(x)
    stopifnot(n > 1, m == length(target))

    ## set initial values for weights ##
    weight <- sweights/sum(sweights)

    ## set initial values for coefficents ##
    beta <- rep(0, m)

    z <- target

    delta <- rep(99, m)

    it <- 1
    while(max(abs(delta)) >= tol & it < maxiter){
          zz <- apply(x * weight, 2, sum)
          xx <- x - matrix(zz, nrow=n, ncol=m, byrow=T)
          xxw <- xx * weight
          sigma <- t(xxw) %*% xx
          delta <- qr.solve(sigma,z-zz, tol=1e-10)
          w2 <- max(abs(delta))
          if(w2 > maxdelta){ delta <- maxdelta*delta/w2}
          beta <- delta+beta

          weight <- as.numeric(exp(x %*% beta))*sweights;
          weight <- weight/sum(weight);

          it <- it+1
    }

  invsigma <- solve(sigma)
  covbeta <- invsigma %*% (t(xxw) %*% xxw) %*% invsigma
  stdbeta <- sqrt(diag(covbeta))
```

```
    weight <- as.numeric(exp(x %*% beta))
    weight <- weight/sum(weight*sweights)

    res <- list(beta=beta, se_beta=stdbeta, covbeta=covbeta,
       weights=weight)
  }

  res <- try(.mdia(x=x, target=target, sweights=sweights,
     maxdelta=maxdelta, maxiter=maxiter, tol=tol) )
  if(is.null(attr(res, "condition"))){
     if(any(abs(apply(x*res$weights*sweights, 2, sum)-target) >
        tol)){res$exitcode <- -2}else{
     res$exitcode <- 0 }
  }else{
  res <- list(beta=NULL, se_beta=NULL, covbeta=NULL, weights=NULL,
     exitcode=-1)
  }

  return(res)

}

### Entropy Balance estimation
### Wrap to include this function from the data directly
### data = dataframe containing variables
### treatment = treatment variable
### covariates = covariates to be balanced
mdiawrap <- function(data=dat,
                     treatment="z.a",
                     covariates = covariatelist[[1]],
                     maxdelta=1, maxiter=1000, tol=1e-08){
  mytar <- sapply(data[data[,treatment] == 1, covariates], mean)
  mywgt <- mdia(data[data[,treatment] == 0, covariates], target=mytar,
     maxdelta=maxdelta, maxiter=maxiter, tol=tol)
  wt <-rep(1,dim(data)[1])
  if(mywgt$exitcode ==0){ wt[data[,treatment] == 0] <- mywgt$weights}
  if(mywgt$exitcode !=0){ wt <-rep(NA,dim(data)[1])}
  return(wt)
}

F.hain.sim <- function(dataset,assign,version) {
  return.obj <-  mdiawrap(data=dataset, treatment=assign, covariates =
     covariatelist[[version]])
  return(return.obj)
}



## a function to check the method needed

methcheck <- function(cmethods=cmethods,value = "lrg"){
  returnval <- 0
  for(val in cmethods) {
   if(tolower(val) == tolower(value)) returnval <- 1
```

```
  }
return(returnval)
 }

#######################################################
#CALLS
#######################################################

#schema list
#schema list
## This is ultimaley setp in the SimulationRun.R file
#scenarios <- c("a","e","g")
#versions <- 1:6

## Set up different output names using the variable "claude"
##    in SimulationRun.R
f <- paste(dat.dir,"image_",claude,".RData",sep="")
ff <- paste(dat.dir,"Vimage_",claude,".RData",sep="")


#Logistic weights
add.lrg <- function(sim) {
  for(letter in scenarios) {
    for(version in versions) {
    assign <- paste("z.",letter,sep="")
    lrgmod <- F.lrg.sim(sim,assign,version)
    sim[,paste(letter,".",version,".lrg.pscore",sep="")] <-
        lrgmod$pscore
    sim[,paste(letter,".",version,".lrg.att.wt",sep="")] <-
        ifelse(sim[,assign]==1, 1, lrgmod$pscore/(1-lrgmod$pscore))
  }
  }
  save.image(ff)
  return(sim)
}


#Run a particular method if it is in teh CMETHODS
mycheck <- methcheck(cmethods = cmethods, value="lrg")
if(mycheck == 1){ simdata <- lapply(simdata,add.lrg)  }
save.image(f)

#GBM weights
add.gbm <- function(sim) {
  for(letter in scenarios) {
    for(version in versions) {
    assign <- paste("z.",letter,sep="")
    #calculate the ps
    psm.att <- tryCatch({F.gbm.sim(sim,"ATT",assign,version)},
        error=function(e) list(ps=rep(NA,dim(sim)[1])))
    #augment the dataset
    sim[,paste(letter,".",version,".gbm.att.pscore",sep="")] <-
        psm.att$ps[,1]
```

```
    sim[,paste(letter,".",version,".gbm.att.wt",sep="")] <-
        ifelse(!is.na(psm.att$ps), get.weights(psm.att, estimand="ATT",
        stop.method="ks.mean") , NA)
  }
  }
  return(sim)
}


#Run a particular method if it is in teh CMETHODS
mycheck <- methcheck(cmethods = cmethods, value="gbm")
if(mycheck == 1){ simdata <- lapply(simdata,add.gbm)  }
save.image(f)



#CBPS weights
add.cbps <- function(sim) {
  for(letter in scenarios) {
    for(version in versions) {
    assign <- paste("z.",letter,sep="")
    #calculate the ps
    cbpsm.att <- tryCatch({F.cbps.sim(sim,"ATT",assign,version)},
        error=function(e) list(fitted.values=rep(NA,dim(sim)[1])))
    #augment the dataset
    sim[,paste(letter,".",version,".cbps.att.pscore",sep="")] <-
        cbpsm.att$fitted.values
    sim[,paste(letter,".",version,".cbps.att.wt",sep="")] <-
        ifelse(!is.na(cbpsm.att$fitted.values), ifelse(sim[,assign]==1,
        1, cbpsm.att$fitted.values/(1-cbpsm.att$fitted.values)) , NA)
  }
  }
  return(sim)
}

#Run a particular method if it is in teh CMETHODS
mycheck <- methcheck(cmethods = cmethods, value="cbps")
if(mycheck == 1){ simdata <- lapply(simdata,add.cbps) }
save.image(f)

#Hainmueller's Entropy balance proposed weights
add.hain <- function(sim) {
  for(letter in scenarios) {
    for(version in versions) {
    assign <- paste("z.",letter,sep="")
    #calculate the ps
    hainwt.att <- F.hain.sim(sim,assign,version)
    #augment the dataset
    sim[,paste(letter,".",version,".hain.att.wt",sep="")] <- hainwt.att
  }
  }
  return(sim)
}
```

```
#Run a particular method if it is in the CMETHODS
mycheck <- methcheck(cmethods = cmethods, value="hain")
if(mycheck == 1){ simdata <- lapply(simdata,add.hain) }
#save.image(f)

## Save all data for future additional processing when needed

f <- paste(dat.dir,"image_",claude,".RData",sep="")
save.image(f)
```

7.4 Summary functions in the file "FSummary.R"

```
library(survey)

########################################
#Summary statistics
########################################

########################################
#ASAM
########################################

#Calculate standardized difference
F.std.diff <- function(u,z,w) {
  if(!is.factor(u)) {
    sd1 <- sd(u[z==1], na.rm=T)
    if(sd1 > 0) {
      result <- abs(mean(u[z==1],na.rm=TRUE)-
                      weighted.mean(u[z==0],w[z==0],na.rm=TRUE))/sd1
    } else {
      result <- 0
      warning("Covariate with standard deviation 0.")
    }
  }
  else {
    result <- NULL
    for(u.level in levels(u)) {
      result <- c(result, std.diff(as.numeric(u==u.level),z,w))
    }
  }
  return(result)
}

#Calculate ASAM for one iteration
calc_asam <- function(sim,assign,weight,covariates) {
  sam <- sapply(covariates,function(covariate)
     {F.std.diff(sim[,covariate],sim[,assign],sim[,weight])})
  asam <- mean(sam)
  return(asam)
}

#Calculate bias for one iteration
calc_bias <- function(sim,assign,outcome,weight, trueval=g1) {
  treated <- sim[,assign]==1
  bias <- weighted.mean(sim[treated,outcome],sim[treated,weight])-
      weighted.mean(sim[!treated,outcome],sim[!treated,weight])- trueval
  return(bias)
}

#Calculate absolute bias for one iteration
calc_absbias <- function(sim,assign,outcome,weight, trueval=g1) {
  treated <- sim[,assign]==1
```

```
  absbias <-
     abs((weighted.mean(sim[treated,outcome],sim[treated,weight])-
            weighted.mean(sim[!treated,outcome],sim[!treated,weight])-
            trueval)/trueval)
  return(absbias)
}

#Calculate Mean Square Error MSE for one iteration
calc_mse <- function(sim,assign,outcome,weight, trueval=g1) {
  treated <- sim[,assign]==1
  mse <- (weighted.mean(sim[treated,outcome],sim[treated,weight])-
          weighted.mean(sim[!treated,outcome],sim[!treated,weight])-
          trueval)^2
  return(mse)
}

#Calculate Mean Square Error MSE for one iteration
calc_sqbias <- function(sim,assign,outcome,weight, trueval=g1) {
  treated <- sim[,assign]==1
  sqbias <- (weighted.mean(sim[treated,outcome],sim[treated,weight])-
            weighted.mean(sim[!treated,outcome],sim[!treated,weight])-
            trueval)^2
  return(sqbias)
}


#Calculate SE for one iteration
calc_se <- function(sim,assign,outcome,weight) {
data <- data.frame(assign=sim[,assign],
                   outcome=sim[,outcome],
                   weight=sim[,weight])
  nodesign <- svydesign(ids=~1,weights=data$weight,data=data)
  x <- svyglm(outcome~assign,design=nodesign,data=data)
  se <- summary(x)$coefficients[2,2]
  return(se)
}

#Calculate KS statistics for one iteration

ks.est<-function (x, t, w, get.means = FALSE, get.ks = TRUE,
                  na.action = c("level", "exclude", "lowest")[1],
                  collapse.by.var = FALSE)
{
  design <- svydesign(ids = ~1, weights = ~w,
                      data = data.frame(x = x,
                      t = t, w = w, miss = is.na(x)))
  if (na.action == "exclude")
    design <- subset(design, !is.na(x))
  ret <- NULL
  if (get.means) {
    design.t <- subset(design, t == 1)
    design.c <- subset(design, t == 0)
    m.t <- svymean(~x, design.t, na.rm = TRUE)
    m.c <- svymean(~x, design.c, na.rm = TRUE)
```

```
    sd.t <- sqrt(svyvar(~x, design.t, na.rm = TRUE))
    sd.c <- sqrt(svyvar(~x, design.c, na.rm = TRUE))
    t.n <- summary(svyglm(x ~ t, design))$coefficients[2,3:4]
    ####################
    #new denom for ATE - do I need tildas?
    ####################
    m.t.unwt=mean(x[t==1],na.rm=TRUE)
    m.c.unwt=mean(x[t==0],na.rm=TRUE)
    dev.t<-sum((x[t==1]-m.t.unwt)^2)
    dev.c<-sum((x[t==0]-m.c.unwt)^2)
    N.t=length(x[t==1])
    N.c=length(x[t==0])
    sd.p=sqrt((dev.t+dev.c)/(N.t+N.c-2))
    b.n <- ifelse(sd.t == 0, NA, (m.t - m.c)/sd.p)
    #######
    ret <- cbind(m.t, sd.t, m.c, sd.c, b.n, t.n[1], t.n[2])
    colnames(ret) <- c("tx.mn", "tx.sd", "ct.mn", "ct.sd",
                       "std.eff.sz", "stat", "p")
    if ((sum(is.na(x)) > 0) && (na.action == "level")) {
      m.t <- svymean(~is.na(x), design.t, na.rm = TRUE)[2]
      m.c <- svymean(~is.na(x), design.c, na.rm = TRUE)[2]
      sd.t <- sqrt(m.t * (1 - m.t))
      sd.c <- sqrt(m.c * (1 - m.c))
      test <- try(summary(svyglm(is.na(x) ~ t, family = quasibinomial,
                                 design)), silent = TRUE)
      if (class(test)[1] != "try-error")
        t.n <- test$coefficients[2, 3:4]
      else t.n <- c(NA, NA)
      b.n <- ifelse(sd.t == 0, NA, (m.t - m.c)/sd.t)
      ret <- rbind(ret, c(m.t, sd.t, m.c, sd.c, b.n, t.n))
    }
  }
  if (get.ks) {
    work <- design$variables
    if (na.action == "lowest")
      work$x[is.na(work$x)] <- min(work$x, na.rm = TRUE) - 1
    if (na.action == "level")
      work <- subset(work, !is.na(x))
    work$w[work$t == 1] <- with(subset(work, t == 1), w/sum(w))
    work$w[work$t == 0] <- with(subset(work, t == 0), -w/sum(w))
    ess <- with(work, sapply(split(w, t), function(w) {
      sum(w)^2/sum(w^2)
    }))
    ind <- order(work$x)
    cumv <- abs(cumsum(work$w[ind]))
    cumv <- cumv[diff(work$x[ind]) != 0]
    ks <- ifelse(length(cumv) > 0, max(cumv), 0)
    pval <- 1 - .C("psmirnov2x", p = as.double(ks), as.integer(ess[2]),
                   as.integer(ess[1]), PACKAGE = "twang")$p
    if ((sum(is.na(design$variables$x)) > 0) && (na.action =="level")) {
      work <- design$variables
      work$w[work$t == 1] <- with(subset(work, t == 1),
                                  w/sum(w))
      work$w[work$t == 0] <- with(subset(work, t == 0),
```

```
                                    -w/sum(w))
      ks <- c(ks, abs(sum(with(subset(work, is.na(x)),
                               sapply(split(w, t), sum)))))
      pval <- c(pval, as.numeric(svychisq(~miss + t, design =
         design)$p.value))
    }
    ret <- cbind(ret, ks, ks.pval = pval)
  }
  ret <- data.frame(ret)
  rownames(ret) <- c("", "<NA>")[1:nrow(ret)]
  return(ret)
}


ks.sum <- function(data=cst, mvars = c("w2","w4","w1"), treat="z.a",
    weight="a.2.cbps.att.pscore" , type="mean")
{
mdat <- data
tread <- lapply(data[, mvars, drop = FALSE],
                        ks.est, t = data[, treat], w = data[, weight])

n.rows <- sapply(tread, nrow)
var.levels <- unlist(sapply(tread, rownames))
var.names <- rep(names(tread), n.rows)
var.names[var.levels != ""] <- paste(var.names[var.levels != ""],
                                var.levels[var.levels != ""], sep = ":")
rest <- data.frame(matrix(0, nrow = length(var.names),
                    ncol = ncol(tread[[1]])))
names(rest) <- colnames(tread[[1]])
rownames(rest) <- var.names

i.insert <- 1
for (i in 1:length(tread)) {
  rest[i.insert:(i.insert + nrow(tread[[i]]) - 1), ] <- tread[[i]]
  i.insert <- i.insert + nrow(tread[[i]])
}
#rest <- list(results = data.frame(rest))
rest <- data.frame(rest)
#rest$names <- row.names(rest)
if(type == "mean"){finalks <- mean(rest[,"ks"])}
if(type == "max"){finalks <- max(rest[,"ks"])}
if(type == "min"){finalks <- min(rest[,"ks"])}

#return(rest)
return(finalks)
}



########################################
#General plots and data estimation for ASAM, BIAS, ABSBIAS, SQBIAS MSE
    SE
########################################
```

```
plot_general <- function(scenario, estimation,estpop, claude=claude,
    covariatelist=covariatelist, general="bias", named="bias") {
  zname <- paste("z.",scenario,sep="")
  yname <- paste("y.",scenario,sep="")
  versions <- 1:length(covariatelist)
    xx0 <- NULL
  for(ii in 1:length(covariatelist)) {
     if(tolower(general) == "bias") {
     x1 <- sapply(1:nsim,function(k) { calc_bias(simdata[[k]], zname,
            yname, paste(scenario,ii,estimation,estpop,"wt",sep="."))})}
     if(tolower(general) == "absbias") {
     x1 <- sapply(1:nsim,function(k) {calc_absbias(simdata[[k]],zname,
            yname,paste(scenario,ii,estimation,estpop,"wt",sep="."))})}
     if(tolower(general) == "asam") {
     x1 <- sapply(1:nsim,function(k) {calc_asam(simdata[[k]],zname,
            paste(scenario,ii,estimation,estpop,"wt",sep="."),
            covariatelist[[ii]])})}
     if(tolower(general) == "ks") {
     x1 <- sapply(1:nsim,function(k) {ks.sum(data=simdata[[k]],
            mvars=covariatelist[[ii]], treat=zname,
            weight= paste(scenario,ii,estimation,estpop,"wt",sep="."))
               })}
     if(tolower(general) == "ksmax") {
     x1 <- sapply(1:nsim,function(k) {ks.sum(data=simdata[[k]],
            mvars=covariatelist[[ii]], treat=zname,
            weight= paste(scenario,ii,estimation,estpop,"wt",sep="."),
            type="max") })}
     if(tolower(general) == "ksmin") {
     x1 <- sapply(1:nsim,function(k) {ks.sum(data=simdata[[k]],
            mvars=covariatelist[[ii]], treat=zname,
            weight= paste(scenario,ii,estimation,estpop,"wt",sep="."),
            type="min") })}
     if(tolower(general) == "mse") {
     x1 <- sapply(1:nsim,function(k) {calc_mse(simdata[[k]],zname,
            yname,paste(scenario,ii,estimation,estpop,"wt",sep="."))})}
     if(tolower(general) == "sqbias") {
     x1 <- sapply(1:nsim,function(k) {calc_sqbias(simdata[[k]],zname,
            yname,paste(scenario,ii,estimation,estpop,"wt",sep="."))})}
     if(tolower(general) == "se") {
     x1 <- sapply(1:nsim,function(k) {calc_se(simdata[[k]],zname,
            yname,paste(scenario,ii,estimation,estpop,"wt",sep="."))})}

    xx0 <- c(xx0, x1)
       }

  tograph <- as.data.frame(cbind(
    xx0,
    sort(rep(1:length(covariatelist),nsim))
  ))
  nm2 <- gsub(" ","", named , fixed=TRUE)
  names(tograph) <- c("SVAR","version")
  tograph$version <- as.factor(tograph$version)
  tograph$SVAR <- as.numeric(as.character((tograph$SVAR)))
  f <- paste(out.dir,nm2,".",scenario,".",estimation,".",estpop,".",
```

```
             claude,".pdf",sep="")
  graphics.off()
  pdf(f)
  boxplot(SVAR~version,data=tograph,horizontal=T, main=paste(named,",
          scenario", scenario, ",", "pscore type", estimation, ",",
          estpop),ylab="Version")
dev.off()
  names(tograph) <- c(nm2,"version")
  toreturn <- tograph
  toreturn$scenario = scenario
  toreturn$estimation = estimation
  toreturn$estpop = estpop

  return(toreturn)
}




dat_general <- function(scen=scenarios, cmeth =cmethods, estpop="att",
                        claude=claude,  covariatelist=covariatelist,
                        general="bias", named=NULL ){
if (is.null(named)){named <- general}
gendat <- NULL
  for(letter in scen) {
    for(meth in cmeth) {
    mdat0 <- NULL
    mdat0 <- plot_general(scenario=letter,estimation=tolower(meth),
                  estpop=estpop, claude=claude,
                  covariatelist=covariatelist, general=general,
                  named=named)
    gendat <- rbind(gendat, mdat0)
      }
    }
  return(gendat)
}


sum_general <- function(sscen=scenarios, scmeth =cmethods,
                        sestpop="att", sclaude=claude,
                        scovariatelist=covariatelist, sgeneral="bias",
                        snamed="bias" ){

souttab <- NULL
nametab <- NULL

  for(sk in 1:length(sgeneral) ) {
   kgeneral <-sgeneral[sk]
   knamed <- snamed[sk]
   sfdat <- NULL
   sfdat <- dat_general(scen=sscen, cmeth =scmeth, estpop=sestpop,
                    claude=sclaude,  covariatelist=scovariatelist,
                    general=kgeneral, named=knamed )
   nm3 <- gsub(" ","", knamed , fixed=TRUE)
```

```
   srval <- sfdat[, nm3]
   x0 <- aggregate(sfdat[, nm3] ,by=list(sfdat[, "scenario"],
             sfdat[, "version"] , sfdat[, "estimation"]),mean)
   names(x0) <- c("scenario","version","estimation", paste(nm3,"mean",
      sep="."))
   x1 <- aggregate(sfdat[, nm3] ,by=list(sfdat[, "scenario"],
             sfdat[, "version"] , sfdat[, "estimation"]),median)
   names(x1) <- c("scenario","version","estimation",
      paste(nm3,"median", sep="."))
   x2 <- aggregate(sfdat[, nm3] ,by=list(sfdat[, "scenario"],
             sfdat[, "version"] , sfdat[, "estimation"]),
             function(vec) quantile(vec,0.025, na.rm=TRUE) )
   names(x2) <- c("scenario","version","estimation", paste(nm3,"lb",
      sep="."))
   x3 <- aggregate(sfdat[, nm3] ,by=list(sfdat[, "scenario"],
             sfdat[, "version"] , sfdat[, "estimation"]),
             function(vec) quantile(vec,0.975, na.rm=TRUE) )
   names(x3) <- c("scenario","version","estimation", paste(nm3,"ub",
      sep="."))
   if (sk == 1){souttab <- x0[, 1:3]
                nametab <- c("scenario","version","estimation")
             }
   souttab <- cbind(souttab, x0[,4], x1[,4],x2[,4],x3[,4])
   nametab <- c(nametab, paste(nm3,"mean", sep="."),
      paste(nm3,"median", sep="."),
               paste(nm3,"lb", sep="."), paste(nm3,"ub", sep=".") )
   names(souttab) <- nametab

  }
  return(souttab)

}


abd <- sum_general(sscen=scenarios, scmeth =cmethods, sestpop="att",
    sclaude=claude,
                   scovariatelist=covariatelist,
                      sgeneral=c("bias","asam","absbias", "sqbias",
                      "se", "ks", "ksmin", "ksmax"),
                   snamed=c("bias","asam","Absolute bias", "Square
                      Bias", "SE", "KS Mean", "KS Min", "KS Max") )



f <- paste(out.dir,"SummaryTab", claude, ".csv",sep="")
write.csv(abd,f)

##
## Now create the MSE an the small version of the output
##


abc <- abd
abc$mbias <-abs(abc$bias.mean)
```

```
abc$mse <-abs(abc$SquareBias.mean)
abc$rmse <- sqrt(abc$mse)
abc$var <- abc$mse - (abc$mbias)^2
abc$stderr <- sqrt(abc$var)
abc$stderr10 <- 10*abc$stderr

abc$bias10 <- abc$bias.mean *10
abc$biaslb10 <- abc$bias.lb *10
abc$biasub10 <- abc$bias.ub *10

abc$mbias100 <- abc$mbias *100
abc$mbiaslb100 <- abs(abc$bias.lb)
abc$mbiasub100 <- abs(abc$bias.ub)
abc$mse100 <-100*abs(abc$SquareBias.mean)
abc$mselb100 <-100*abs(abc$SquareBias.lb)
abc$mseub100 <-100*abs(abc$SquareBias.ub)
abc$var100 <- 100*abc$var
abc$stderr10 <- 10*abc$stderr

abe <- abc[,c("scenario", "version", "estimation","bias10", "mse100",
    "var100", "asam.mean", "KSMean.mean", "KSMin.mean", "KSMax.mean")]

f2 <- paste(out.dir,"RestrictedTab", claude, ".csv",sep="")
write.csv(abe,f2)







################################################
#Balance plot
################################################

plot_balance_1v <- function(scenario,version,estimation,estpop,i) {
sim <- simdata[[i]]
treat <- sim[,paste("z.",scenario,sep="")]
weight <- sim[,paste(scenario,version,estimation,estpop,"wt",sep=".")]
noweight <- rep(1,nobs)
covariates <- covariatelist[[version]]
ncov <- length(covariates)

x1 <- sapply(covariates,function(c) F.std.diff(sim[,c],treat,weight))
x2 <- sapply(covariates,function(c) F.std.diff(sim[,c],treat,noweight))
labels <- covariates

y <- rep(1:ncov,2)
x <- c(x1,x2)
ylim <- c(0.5,ncov+1.5)
#range <- max(x1) - min(x1)
#xlim <- c(min(x)-range/5,max(x))
colors <- c(rep("blue",ncov),rep("red",ncov))
plot(x,y,col=colors,pch=19,ylim=ylim,yaxt='n',ylab="Covariate",
```

```
     xlab="Standardized difference", main=paste("Version", version))
axis(2,at=1:ncov,labels=covariates)
for(i in 1:ncov) abline(h=i,col="grey")
abline(v=0,col="grey")
legend(x=min(x),y=ncov+1.75,horiz=T,cex=0.8,
    legend=c("Unweighted","Weighted"),
    fill=c("red","blue"),border="white",box.col="white")
}

plot_balance <- function(scenario,estimation,estpop,i, claude=claude,
    covariatelist=covariatelist) {
  f <- paste(out.dir,"balance.",scenario,".",estimation,".",estpop,
             ".",claude,".pdf",sep="")
  graphics.off()
  pdf(f)
par(mfrow=c(2,3))
  for(ii in 1:length(covariatelist)) {
    plot_balance_1v(scenario,ii,estimation,estpop,i)
       }
dev.off()
}



  for(letter in scenarios) {
    for(meth in cmethods) {
     plot_balance(scenario=letter,estimation=tolower(meth),estpop="att",
                  i=1, claude=claude,  covariatelist=covariatelist)
      }
    }
```

7.5 The SAS code"addoutcomes.sas" to generate additional outcomes

```
/*
Author: Claude Setodji.
This SAS code is written to complement the R code used for the estimation of the
    propensity score.
The data from R is imported here and different outcome models are created using this code.
Propensity score weighted estimates are also computed
*/
%let dir=/vincent/b/setodji/causal;;



libname dir "&dir";

%macro _counts_(arg1, text=&arg1, notes=, split=%str( ));

%local i;
%let i=0;

%do %while(%length(%nrbquote(%scan(&text, &i+1, &split))));
       %let i=%eval(&i+1);
%end;

&i
```

```
%if %length(&notes) %then %put NOTE: _COUNT is returning the value: &i..;
%mend _counts_;


%macro covariatelist;
w1 w2 w3 w4 -
w1 w2 w3 w4 w5 w6 w7 -
w1 w2 w3 w4 w8 w9 w10 -
w1 w2 w3 w4 w5 w6 w8 w9 -
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 -
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12 w13 w14 w15
%mend;


%macro addlist;
w1 w2 w3 w4 w2a w2b w4a w4b w24 -
w1 w2 w3 w4 w5 w6 w7 w2a w2b w4a w4b w7a w7b w24 w57-
w1 w2 w3 w4 w8 w9 w10 w2a w2b w4a w4b w10a w10b w24-
w1 w2 w3 w4 w5 w6 w8 w9 w2a w2b w4a w4b w24 w57-
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w2a w2b w4a w4b w7a w7b w10a w10b w24 w57-
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12 w13 w14 w15 w2a w2b w4a w4b w7a w7b w10a w10b w24
    w11a w11b w12a w12b w57
%mend;



%macro therun(data=temp,
              cov1=%covariatelist,
              cov2=%addlist,
                          id = dataid ,
                          effect= -0.4,
                          error = 1 to 5,
                          errorscale= 4 ,
                          mod=2,
                          estimation = gbm cbps lrg,
                          scenarios = b c d g,
                          out=fin
              );
%local i j a0 a1 a2 a3 a4 a5 a6 a7 g1 scen est mz my mw mx1 mx2 estcheck bb bb0 bb1;
%let a0 = -3.85;
%let a1 = 0.3;
%let a2 = -0.36;
%let a3 = -0.73;
%let a4 = -0.2;
%let a5 = 0.71;
%let a6 = -0.19;
%let a7 = 0.26;
%let g1 = -0.4;
data &out _rho_ _yrho_;
run;
proc sort data=&data;
 by &id;
run;

/*Estimate the treatment logistic model Adjusted R-square;;*/
ods listing close;
%do i=1 %to %_counts_(&cov1, split=-);
 %let my = ;
 %let mw = ;
 %let my = %scan(&cov1, &i, "-");
 %if &i le %_counts_(&cov2, split=-) %then %let mw = %scan(&cov2, &i, "-");
  %do j= 1 %to %_counts_(&scenarios);
   %let scen= %scan(&scenarios, &j);
   %let mz = ;
   %let mz = z_&scen;
   data _ad_ _ab_ _ac_;
   run;
```

```
   proc logistic data=&data;
    by &id;
        model &mz = &my /rsq ;;;;
        ods output rsquare=_ab_(keep=&id cvalue1 rename=(cvalue1= Rsq_z1))
                   association=_ac_(where=(label2="c") keep=&id nvalue2 label2
                       rename=(nvalue2= cstat_z1));;;;
   run;;;
   data _ad_;
    length scenario $256;;
    merge _ab_ _ac_(drop=label2);
        by &id;
        scenario="&scen";
        version=&i;
   run;
   %if &mw ^=%str() %then %do;
   data _ab2_ _ac2_;
   run;
   proc logistic data=&data;
    by &id;
        model &mz = &mw /rsq ;
        ods output rsquare=_ab2_(keep=&id cvalue1 rename=(cvalue1= Rsq_z2))
                   association=_ac2_(where=(label2="c") keep=&id nvalue2 label2
                       rename=(nvalue2= cstat_z2));
   run;
   data _ad_;
    merge _ad_ _ab2_ _ac2_(drop=label2);
        by &id;
        RsqDiff_z = Rsq_z2 -Rsq_z1;
        CstatDiff_z = cstat_z2 - cstat_z1;
   run;
   %end;
  data _rho_;
   set _rho_ _ad_;
   if version=. then delete;
  run;
  %end;


%end;
ods listing;


%do i1= %scan(&error, 1) %to %scan(&error, 3);
  %do model =1 %to &mod;

  data stemp;
  run;
  data stemp;
   set &data;
   %if &model = 1 %then az = &a0 + &a1*w1 + &a2*w2*w1 + &a3*w3 + &a4*w4 + &a5*w8*w3 +
        &a6*w9 + &a7*w10*w4 ;;;
   %if &model = 2 %then az = &a0 + &a1*w1 + &a2*w8*(w2**2) + &a3*w3 + &a4*w1*exp(1.3*w4)
        +&a5*w8 + &a6*w9 + &a7*w10 ;;;
   %if &model = 3 %then az = &a0 + exp(&a1*w1 + &a2*w2 + &a3*w1*w3 + &a4*w4 +&a5*w8 +
        &a6*w9 + &a7*w8*w10);;;
   %if &model = 4 %then az = &a0 + w8*exp(&a1*w1 + &a2*(w2**2) + &a3*w3) +
        &a4*exp(1.3*w4) +&a5*w8 + &a6*w1*w9 + &a7*w10;;;
   %if &model = 5 %then az = &a0 + 4*sin(&a1*w1 + &a2*w2*w1 + &a3*w3 + &a4*w4) +
        &a5*w8*w3 + 2*sin(&a6*w9 + &a7*w10*w4) ;;;
   %if &model = 6 %then az = &a0 + 4*sin(&a1*w1 + &a2*w2*w4 + &a3*w3 + &a4*w1*w8 +&a5*w8
        + &a6*w9 + &a7*w10);;;
   y_a = az + (&i1/&errorscale)*rannor(123&i1&model) + (&effect*z_a);
   y_b = az + (&i1/&errorscale)*rannor(124&i1&model) + (&effect*z_b);
   y_c = az + (&i1/&errorscale)*rannor(125&i1&model) + (&effect*z_c);
   y_d = az + (&i1/&errorscale)*rannor(126&i1&model) + (&effect*z_d);
   y_e = az + (&i1/&errorscale)*rannor(127&i1&model) + (&effect*z_e);
   y_f = az + (&i1/&errorscale)*rannor(128&i1&model) + (&effect*z_f);
   y_g = az + (&i1/&errorscale)*rannor(129&i1&model) + (&effect*z_g);
```

```
 y_h = az + (&i1/&errorscale)*rannor(133&i1&model) + (&effect*z_h);
 y_i = az + (&i1/&errorscale)*rannor(143&i1&model) + (&effect*z_i);
 y_j = az + (&i1/&errorscale)*rannor(153&i1&model) + (&effect*z_j);
run;

%do i2= 1 %to %_counts_(&scenarios);
 %let scen=;
 %let scen=%scan(&scenarios, &i2);
 %let mz = ; %let my = ;
              %let my = y_&scen;
              %let mz = z_&scen;
       %do version =1 %to %_counts_(&cov1, split=-);;
       /* First estimate the outcome correlations;*/
       %let mx1 = ;
       %let mx2 = ;
       %let mx1 = %scan(&cov1, &version, "-");;
       %if &version le %_counts_(&cov2, split=-) %then %let mx2 = %scan(&cov2,
           &version, "-");
       data _ab_;
       run;
       proc reg data=stemp outest=_ab_(keep=&id _rsq_ rename=(_rsq_=Rsq_y1)) rsquare
           noprint;
        by &id;
        model &my = &mx1 &mz ;
       run;
       data _ab_;
        length scenario $256;
        set _ab_;
        version=&version;
        scenario="&scen";
        model= &model;
        error=&i1/&errorscale;
       run;
       %if &mx2 ^=%str() %then %do;
       data _ac_;
       run;
       proc reg data=stemp outest=_ac_(keep=&id _rsq_ rename=(_rsq_=Rsq_y2)) rsquare
           noprint;
        by &id;
        model &my = &mx2 &mz ;
       run;
       data _ab_;
        merge _ab_ _ac_;
        by &id;
        RsqDiff_y = Rsq_y2 -Rsq_y1;
       run;
       %end;
       data _yrho_;
        set _yrho_ _ab_;
        if version=. then delete;
       run;
       /* end of outcome correlation estimation;*/
       %let estcheck = 0;
       data _ca_;
       run;
         %do i3= 1 %to %_counts_(&estimation);
          %let est=;
     %let est=%scan(&estimation, &i3);
              %let mw = ;
              %let mw =&scen._&version._&est._att_wt;
   data _a_ _b_ _bb_;
       run;
       proc means data =stemp noprint;
         class &mz;
             var   &my;
             by &id;
             output out=_b_ std=std;
       run;
```

```
            data _b_;
            merge _b_(where=(&mz = .))
              _b_(where=(&mz = 0) rename=(std=std0))
                          _b_(where=(&mz = 1) rename=(std=std1));
                by &id ;
                drop &mz _type_;
                run;
/** Now estimate the standard deviations at the population level , i.e. across MC
    simulations;;*/
          proc means data =stemp noprint;
            class &mz;
                var   &my;
                output out=_bb_ std=std;
          run;

%let bb=0;
%let bb0=0;
%let bb1=0;

data _bb_;
 set _bb_;
 if &mz = . then call symput("bb", put(std, best12.));
 if &mz = 0 then call symput("bb0", put(std, best12.));
 if &mz = 1 then call symput("bb1", put(std, best12.));
run;

data _b_;
 set _b_;
 MCstd = &bb;
 MCstd0 = &bb0;
 MCstd1 = &bb1;
run;

      proc means data =stemp noprint;
            class &mz;
                var   &my;
                weight &mw;
                by &id;
                output out=_a_(where=(_type_=1)) mean=mean;
          run;
          data _a_;
           length scenario estimation $256;
           merge _a_(where=(&mz =1) rename=(mean=mean1)) _a_(where=(&mz =0)
               rename=(mean=mean0)) _b_;;
           by &id;
           effect =mean1 -mean0;
           bias= effect - &effect;
           sqbias=bias**2;
           version=&version;
           scenario="&scen";
           estimation="&est";
           model= &model;
           error=&i1/&errorscale;
           stval=1;
           if effect = . then call symput("estcheck", stval);
           drop _type_ _freq_ &mz stval;
          run;
          %let estcheck = %eval(&estcheck + 0);
          data _ca_;
           set _ca_ _a_;
           if version=. then delete;
          run;
          %put error &i1, estimation &est, scenario &scen, version &version, model
              &model, estcheck &estcheck ;;
      %end;
          %if &estcheck = 1 %then %do;
          data _ca2_;
```

```
          run;
          data _ca2_;
           set _ca_;
           if effect = .;
           keep &id;;
          run;
          proc sort data= _ca2_ nodupkey;
           by &id;
          run;
          proc sort data= _ca_;
           by &id;
          run;
          data _ca_;
           merge _ca_ _ca2_(in=a);
           by &id;
           if a then delete;
          run;
          %end;
          data &out;
           set &out _ca_;
           if version=. then delete;
          run;
         %end;
  %end;

  %end;
%end;

proc sort data=&out;
 by &id version scenario model error estimation;
run;
proc sort data= _rho_;
 by &id version scenario;
run;
proc sort data=_yrho_;
 by &id version scenario model error;
run;
data &out;
 merge &out(in=a) _rho_;
 by &id version scenario;
 if a;
run;
data &out;
 merge &out(in=a) _yrho_;
 by &id version scenario model error;
 if a;
run;


%mend;

/* Import data from the SAS output;
* This data comes out of the "SimulationRun.R" output;
* There are 6 versions of this data and they all should be used but;
** this code is only dealing wiht the first one;*/
 proc import datafile="&dir\image_LeeS1Last1000.csv" out=temp dbms=csv replace;
    getnames=yes;
 run;


data temp;
 set temp;
 w2a = w2**2;
 w2b = w2**3;
 w4a = w4**2;
 w4b = w4**3;
 w7a = w7**2;
 w7b = w7**3;
```

```
 w10a = w10**2;
 w10b = w10**3;
 w11a = w11**2;
 w11b = w11**3;
 w12a = w12**2;
 w12b = w12**3;
 w24=w2*w4;
 w57=w5*w7;
run;


/* This takes a long time to run;
* Preferable split it by the parameter &error;*/
%therun(data=temp,
              cov1=%covariatelist,
              cov2=%addlist,
                          id = dataid ,
                          effect= -0.4,
                          error = 1 to 20,
                          errorscale= 4 ,
                          mod=6,
                          estimation = gbm cbps lrg,
                          scenarios = a b c d e f g ,
                          out=dir.RunEstimates
              );
```

References

[1] Hainmueller J. "Entropy balancing for causal effects: A multivariate reweighting method to produce balanced samples in observational studies." Political Analysis, 2012, 20(1), 25–46.

[2] Setoguchi S. , Schneeweiss S. , Brookhart M. , Glynn R. , and Cook E. , "Evaluating uses of data mining techniques in propensity score estimation: a simulation study." Pharmacoepidemiology and Drug Safety, 2008, 17(6), 546–555.

[3] Wyss R. , Ellis R. , Brookhart A. , Girman C. , Jonsson Funk M. , LoCasale R. , and Sturmer T. , "The role of prediction modeling in propensity score estimation: an evaluation of logistic regression, bcart, and the covariate-balancing propensity score." American Journal of Epidemiology, 2014, 180(6), 645–655.

[4] Lee B. , Lessler J. , and Stuart E. , "Improving propensity score weighting using machine learning." Statistics in Medicine, 2009, 29(3), 337–346.

[5] Cox D. and Snell E. , Analysis of Binary Datas. Boca Raton, FL: Chapman and Hall CRC, 1989.

[6] Maddala G. , Limited Dependent and Qualitative Variables in Econometrics. Cambridge University Press, 1983.