MY457/MY557: Causal Inference for Experimental and Observational Studies

Class 2: Subclassification, Regression, Matching, and Weighting

In the exercise today, we show examples of how the different types of estimators that we discussed in the lectures in weeks 3 and 4 can be implemented in R. This is done using a single simulated dataset, for demonstration purposes.

First, we will load in some required packages:

```r
library(lmtest)
library(sandwich)
library(Matching)
library(PSweight)
library(dplyr)
library(ggplot2)
library(cobalt)
library(knitr)
library(markdown)
library(rlang)
library(tidyverse)
```

Then, we create a simulated dataset. It includes a binary treatment variable D, a group covariate G (with four categories) and two continuous covariates, *X1* and *X2*. The true data-generating mechanism is such that the model for the potential outcomes of Y depends on G, X1 and X2. Similarly, the model for treatment assignment D (i.e. the propensity score) also depends on G, *X1*, *X2*, as well as *X1^2* and .

```r
# GENERATE DATASET

N <- 1000
r12 <- 0.3
treatment_effect <- 5

## a) COVARIATES
X1 <- rnorm(N, mean = 0, sd = 1)
X2 <- rnorm(N, mean = r12*X1, sd = sqrt(1-r12^2))
pG <- exp(cbind(1, X1-X2, X1, 0.5*X1+X2))
pG <- pG/rowSums(pG)
G <- factor(apply(pG, 1 , FUN = function(x){apply(rmultinom(n = 1, size = 1, prob = x)==1, 2, which)}))

## b) POTENTIAL OUTCOMES
Y0 <- rnorm(N, mean = 0*1 + 0.25*as.numeric(G) - 0.5*X1 + (1.25)*X2, sd = 3)
Y1 = Y0 + treatment_effect + rnorm(N, mean = 0, sd = 1)

## c) TREATMENT ASSIGNMENT
pD <- 0.5 + (-0.25)*as.numeric(G) + 0.3*X1 - 0.2*X1^2 - 0.4*X2
pD <- exp(pD)/(1+exp(pD))
D <- factor(rbinom(N, 1, prob = pD))
```

```
## d) ACTUAL OUTCOME
Y <- ifelse(D == 1, Y1, Y0)

###

# COMBINE ALL VARIABLES IN ONE DAATFRAME TOGETHER
df <- data.frame(Y0, Y1, Y, D, G, X1, X2)
```

Study the average treatment effect (ATE)

a) with potential outcomes

If we were to have both potential outcomes, the estimation of the true ATE is easy. We simply take the difference of the means of the potential outcomes, *Y0* and *Y1*.

```
# SIMPLE DIFFERENCE IN MEANS
mean(df$Y1)-mean(df$Y0)
```

```
## [1] 5.016784
```

b) with real outcomes

However, since we do not observe potential outcomes in the real world, we need to work with the real outcome *Y*. If we take the naive approach, unadjusted for any of the covariates, we can get a *biased* estimate using a simple linear regression model:

```
reg.naive <- lm(Y ~ D, data = df)
summary(reg.naive)
```

```
##
## Call:
## lm(formula = Y ~ D, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.4364  -2.2348  -0.0455   2.3401  11.0551
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.7444     0.1409   5.283 1.56e-07 ***
## D1            4.2966     0.2134  20.137  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.346 on 998 degrees of freedom
## Multiple R-squared:  0.2889, Adjusted R-squared:  0.2882
## F-statistic: 405.5 on 1 and 998 DF,  p-value: < 2.2e-16
```

Using control variables

Now, what happens if we control for our observable covariates? Let's run a multiple regression with *Y* as the outcome, *D* ass the treatment assignment indiciator, and *G*, *X1*, and *X2* as control variables. Is the new estimate now closer to the true ATE?

```
reg.1 <- lm(Y ~ D + G + X1 + X2,data = df)
summary(reg.1)
```

```
##
## Call:
## lm(formula = Y ~ D + G + X1 + X2, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.3057  -2.0015   0.0378   1.9999  10.0083
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1022     0.1846   0.554 0.580018
## D1            5.1487     0.2053  25.077  < 2e-16 ***
## G2            0.3792     0.2749   1.379 0.168156
## G3            0.2107     0.3044   0.692 0.488987
## G4            0.8674     0.2857   3.036 0.002463 **
## X1           -0.3801     0.1078  -3.527 0.000439 ***
## X2            1.3762     0.1199  11.476  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.055 on 993 degrees of freedom
## Multiple R-squared:  0.4103, Adjusted R-squared:  0.4067
## F-statistic: 115.1 on 6 and 993 DF,  p-value: < 2.2e-16
```

Subclassification

Next, we calculate subclassification estimates which use G for sub-classification. Remember that subclassification can only be done for categorical variables. To estimate the treatment effect, we needs to follow four steps.

```r
# 1. DEFINE MEAN OUTCOMES FOR TREATMENT AND CONTROL FOR EACH LEVEL OF G
ey11 <- df %>% filter(G == 1 & D == 1) %>% pull(Y) %>% mean(., na.rm = T)
ey10 <- df %>% filter(G == 1 & D == 0) %>% pull(Y) %>% mean(., na.rm = T)
ey21 <- df %>% filter(G == 2 & D == 1) %>% pull(Y) %>% mean(., na.rm = T)
ey20 <- df %>% filter(G == 2 & D == 0) %>% pull(Y) %>% mean(., na.rm = T)
ey31 <- df %>% filter(G == 3 & D == 1) %>% pull(Y) %>% mean(., na.rm = T)
ey30 <- df %>% filter(G == 3 & D == 0) %>% pull(Y) %>% mean(., na.rm = T)
ey41 <- df %>% filter(G == 4 & D == 1) %>% pull(Y) %>% mean(., na.rm = T)
ey40 <- df %>% filter(G == 4 & D == 0) %>% pull(Y) %>% mean(., na.rm = T)

# 2. CALCULATE SIMPLE DIFFERENCE IN MEAN OUTCOMES FOR EACH LEVEL OF G
diff1 = ey11 - ey10
diff2 = ey21 - ey20
diff3 = ey31 - ey30
diff4 = ey41 - ey40

# 3. CALCULATE WEIGHTS FOR EACH LEVEL OF G (GROUP SIZES)
obs = nrow(df %>% filter(D == 0))
wt1 <- df %>% filter(G == 1 & D == 0) %>% nrow(.)/obs
wt2 <- df %>% filter(G == 2 & D == 0) %>% nrow(.)/obs
wt3 <- df %>% filter(G == 3 & D == 0) %>% nrow(.)/obs
wt4 <- df %>% filter(G == 4 & D == 0) %>% nrow(.)/obs

# 4. CALCULATE WEIGHTED AVERAGE TREATMENT EFFECT USING SDOs AND WEIGHTS
watt = diff1*wt1 + diff2*wt2 + diff3*wt3 + diff4*wt4
```

Simple matching

As mentioned above, subclassification is only possible for categorical variables. Thus, if we want to match on continuous variables, we can resort to standard matching approaches. Let's try out some of the most common matching approaches. In the following, I provide four different versions. This enables us to learn more about the different types of matching.

```
# 1. MATCHING: X1 and X2 as variables to match on
match_vars <- model.matrix(~ X1 + X2, data = df)[,-1]
att.m1 <- Match(Y = df$Y, Tr = (df$D==1), X = match_vars, estimand = "ATT", M = 1, replace = TRUE, Weig
summary(att.m1)
```

```
##
## Estimate...  4.8969
## AI SE......  0.2935
## T-stat.....  16.685
## p.val......  < 2.22e-16
##
## Original number of observations..............  1000
## Original number of treated obs...............  436
## Matched number of observations...............  436
## Matched number of observations  (unweighted).  437
```

```
# 2. MATCHING: X1, X2 and G as variables to match on
match_vars <- model.matrix(~ X1 + X2 + G, data = df)[,-1]
att.m2 <- Match(Y = df$Y, Tr = (df$D==1), X = match_vars, estimand = "ATT", M = 1, replace = TRUE, Weig
summary(att.m2)
```

```
##
## Estimate...  5.137
## AI SE......  0.28889
## T-stat.....  17.782
## p.val......  < 2.22e-16
##
## Original number of observations..............  1000
## Original number of treated obs...............  436
## Matched number of observations...............  436
## Matched number of observations  (unweighted).  436
```

```
# 3. MATCHING: X1, X2 and G as variables to match on + inclusion of a regression bias-correction
match_vars <- model.matrix(~ X1 + X2 + G, data = df)[,-1]
att.m3 <- Match(Y = df$Y, Tr = (df$D==1), X = match_vars, estimand = "ATT", M = 1,
                replace = TRUE, Weight = 2, BiasAdjust = T, Z = match_vars)
summary(att.m3)
```

```
##
## Estimate...  5.1957
## AI SE......  0.28881
## T-stat.....  17.99
## p.val......  < 2.22e-16
##
## Original number of observations..............  1000
## Original number of treated obs...............  436
## Matched number of observations...............  436
## Matched number of observations  (unweighted).  436
```

```r
# 4. MATCHING: X1 and X2 as variables to match on + exact matching within G
match_vars <- model.matrix(~ X1 + X2, data = df)[,-1]
att.m4 <- Matchby(Y = df$Y, Tr = (df$D==1), X = match_vars, estimand = "ATT", M = 1,
                  by = df$G, ties = TRUE, replace = TRUE, Weight = 2, AI=TRUE)
```

```
## 1 of 4 groups
## 2 of 4 groups
## 3 of 4 groups
## 4 of 4 groups
```

```r
summary(att.m4)
```

```
##
## Estimate...  5.1464
## AI SE......  0.29162
## AI T-stat..  17.647
## AI p.val...  < 2.22e-16
##
## SE.........  0.21317
## T-stat.....  24.143
## p.val......  < 2.22e-16
##
## Original number of observations..............  1000
## Original number of treated obs...............  436
## Matched number of observations...............  436
## Matched number of observations  (unweighted).  436
```

Propensity score matching

While these simple matching approaches can be very valuable in some settings, we may ran out of observations quite quickly if we add more covariates. To address this *curse of dimensionality*, we can use propensity scores. Using propensity scores for matching consists of three steps:

```r
# 1. ESTIMATE LOGIT/PROBIT MODEL WITH TREATMENT ASSIGNMENT INDICATOR AS OUTCOME
mod.logit <- glm(D ~ G + X1 + I(X1^2) + X2, family = binomial(link = "logit"), data = df)

# 2. CREATE PREDICTION: THESE PREDICTED VALUES ARE YOUR PROPENSITY SCORES
df$prscore <- predict(mod.logit, type = "response")
```

```r
# 3. USE PROPENSITY SCORES FOR MATCHING

# 3a. PS-MATCHING: PS score only
att.pm1 <- Match(Y = df$Y, Tr = (df$D==1), X = df$prscore, estimand = "ATT", M = 1, replace = TRUE, Wei
summary(att.pm1)
```

```
##
## Estimate...  5.0239
## AI SE......  0.30441
## T-stat.....  16.503
## p.val......  < 2.22e-16
##
## Original number of observations..............  1000
## Original number of treated obs...............  436
## Matched number of observations...............  436
## Matched number of observations  (unweighted).  686
```

```r
# 3b. PS-MATCHING: PS score + covariates
match_vars <- model.matrix(~ X1 + X2, data = df)[,-1]
att.pm2 <- Match(Y = df$Y,Tr = (df$D == 1), X = cbind(df$prscore, match_vars), estimand = "ATT", M = 1,
summary(att.pm2)
```

```
##
## Estimate...  5.1066
## AI SE......  0.30223
## T-stat.....  16.896
## p.val......  < 2.22e-16
##
## Original number of observations..............  1000
## Original number of treated obs...............  436
## Matched number of observations...............  436
## Matched number of observations  (unweighted).  437
```

```r
# 3c. PS-MATCHING: PS score + bias-adjustment
match_vars <- model.matrix(~ X1 + X2 + G, data = df)[,-1]
att.pm3 <- Match(Y = df$Y, Tr = (df$D==1), X = df$prscore, estimand = "ATT", M = 1, replace = TRUE, Wei
summary(att.pm3)
```

```
##
## Estimate...  5.0883
## AI SE......  0.29803
## T-stat.....  17.073
## p.val......  < 2.22e-16
##
## Original number of observations..............  1000
## Original number of treated obs...............  436
## Matched number of observations...............  436
## Matched number of observations  (unweighted).  686
```

Diagnostic checks

Finally, the nice thing about matching is that we can assess visually how well the matching (on observables!) worked. To do so, it is common practice to do some diagnostic checks to examine covariate balance before and after matching, and the distribution of the estimated propensity scores.

```r
# 1. BALANCE TABLES
match_vars <- model.matrix(~ X1 + X2 + G, data = df)[,-1]
bal.tab(att.m1, covs = cbind(df$prscore, match_vars), treat = (df$D==1))
```

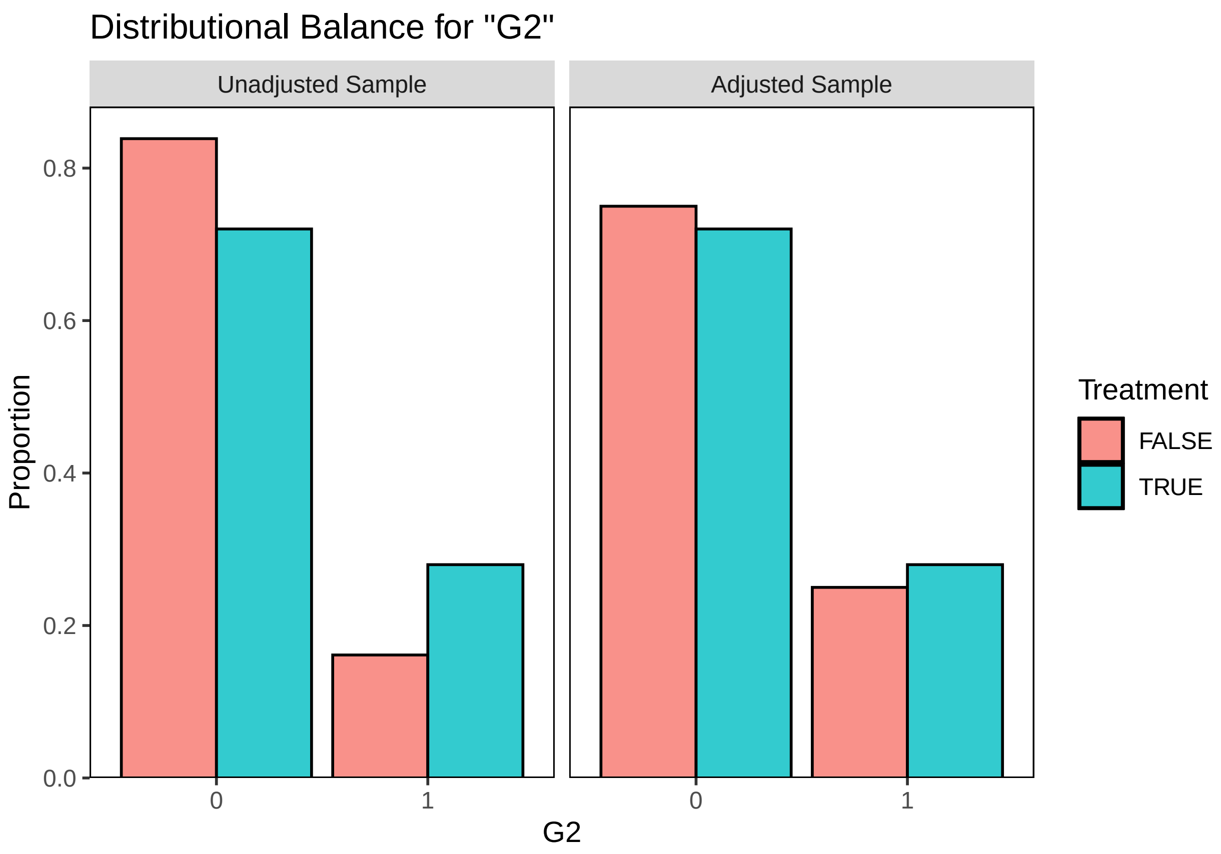
```
## Balance Measures
##       Type Diff.Adj
## V1 Contin.   0.1784
## X1 Contin.  -0.0074
## X2 Contin.  -0.0213
## G2  Binary   0.0298
## G3  Binary  -0.0711
## G4  Binary  -0.1055
##
## Sample sizes
##                      FALSE TRUE
## All                  564.   436
## Matched (ESS)        154.24 436
## Matched (Unweighted) 250.   436
## Unmatched              314.      0
```

```
bal.tab(att.m4, covs = cbind(df$prscore, match_vars), treat = (df$D==1))
```

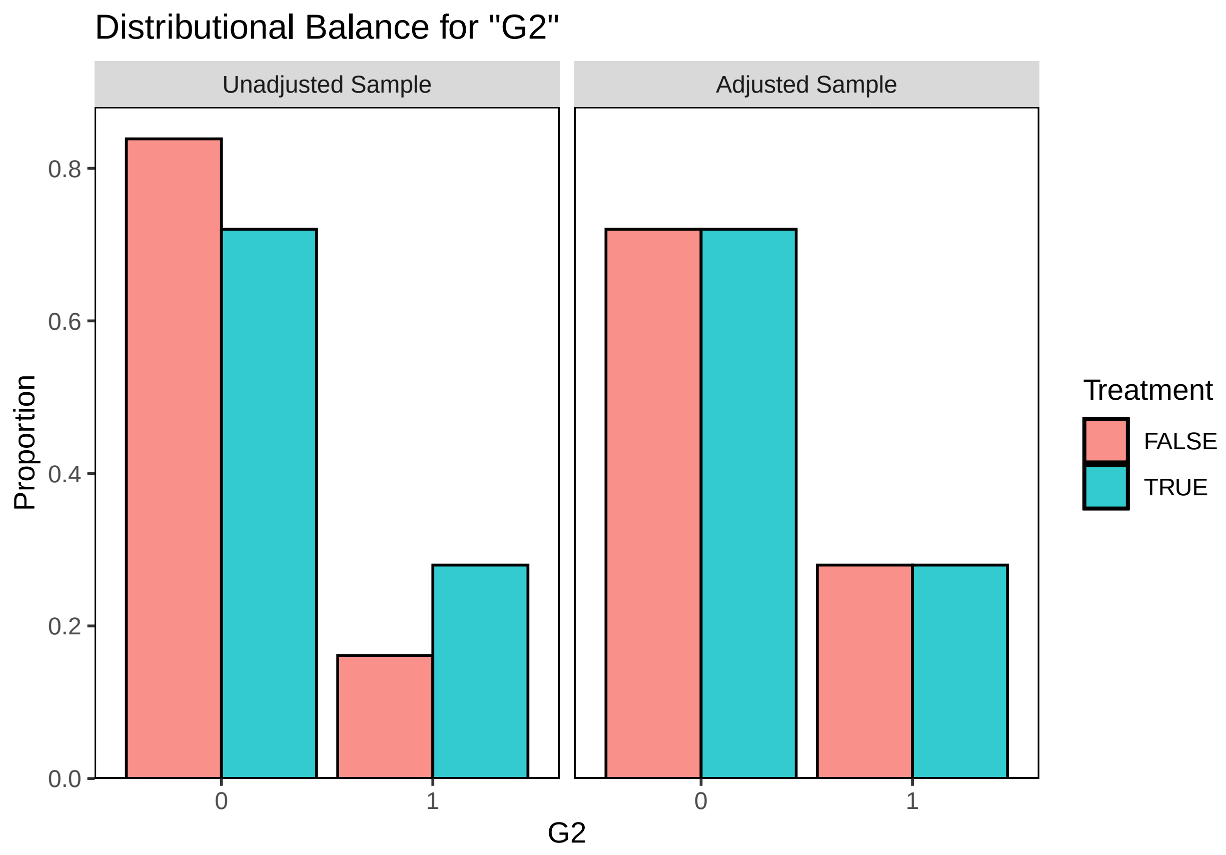
```
## Balance Measures
##       Type Diff.Adj
## V1 Contin.   0.0331
## X1 Contin.   0.0111
## X2 Contin.  -0.0413
## G2  Binary   0.0000
## G3  Binary   0.0000
## G4  Binary   0.0000
##
## Sample sizes
##                      FALSE TRUE
## All                  564.   436
## Matched (ESS)        158.94  436
## Matched (Unweighted) 253.    436
## Unmatched            311.      0
```

```
###

# 2. BALANCE PLOTS
bal.plot(att.m1, which = "both", var.name = "G2", covs = match_vars, treat = (df$D==1))
```

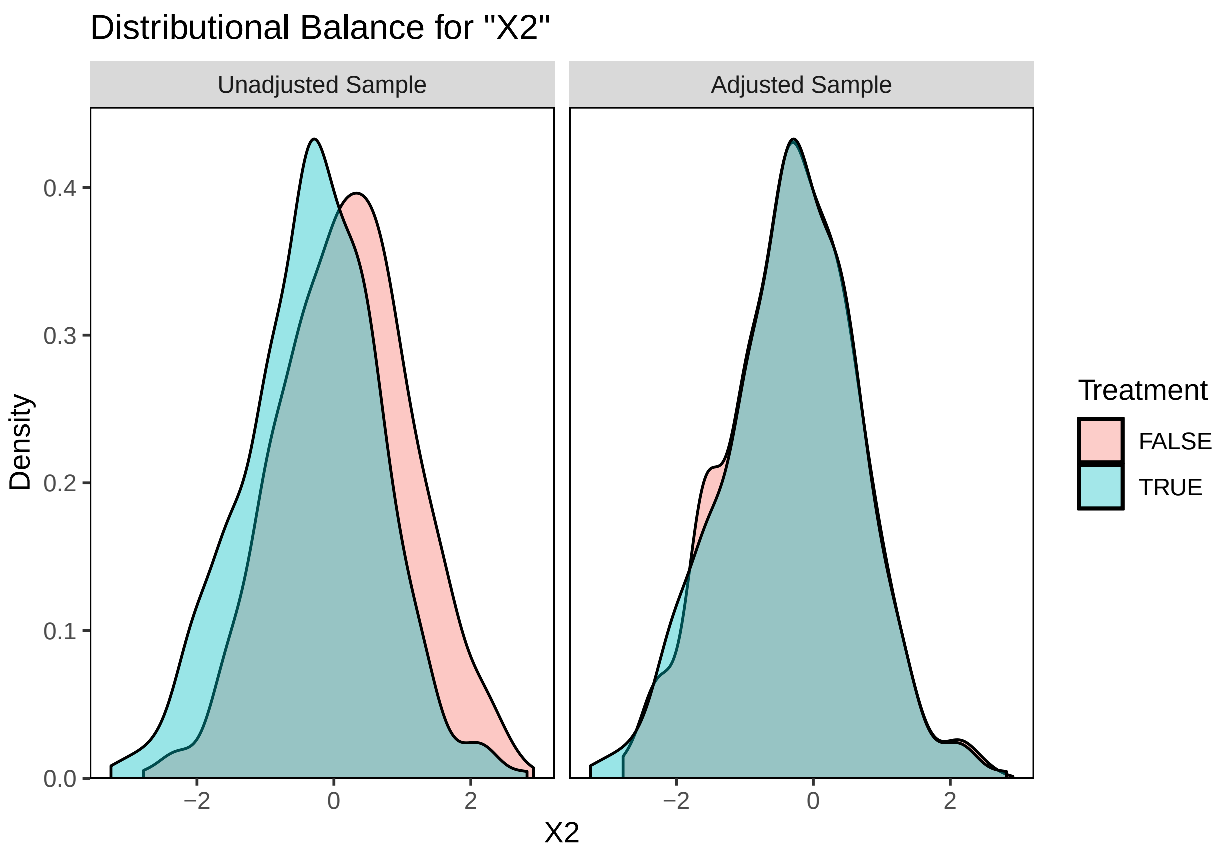


```
bal.plot(att.m4, which = "both", var.name = "G2", covs = match_vars, treat = (df$D==1))
```



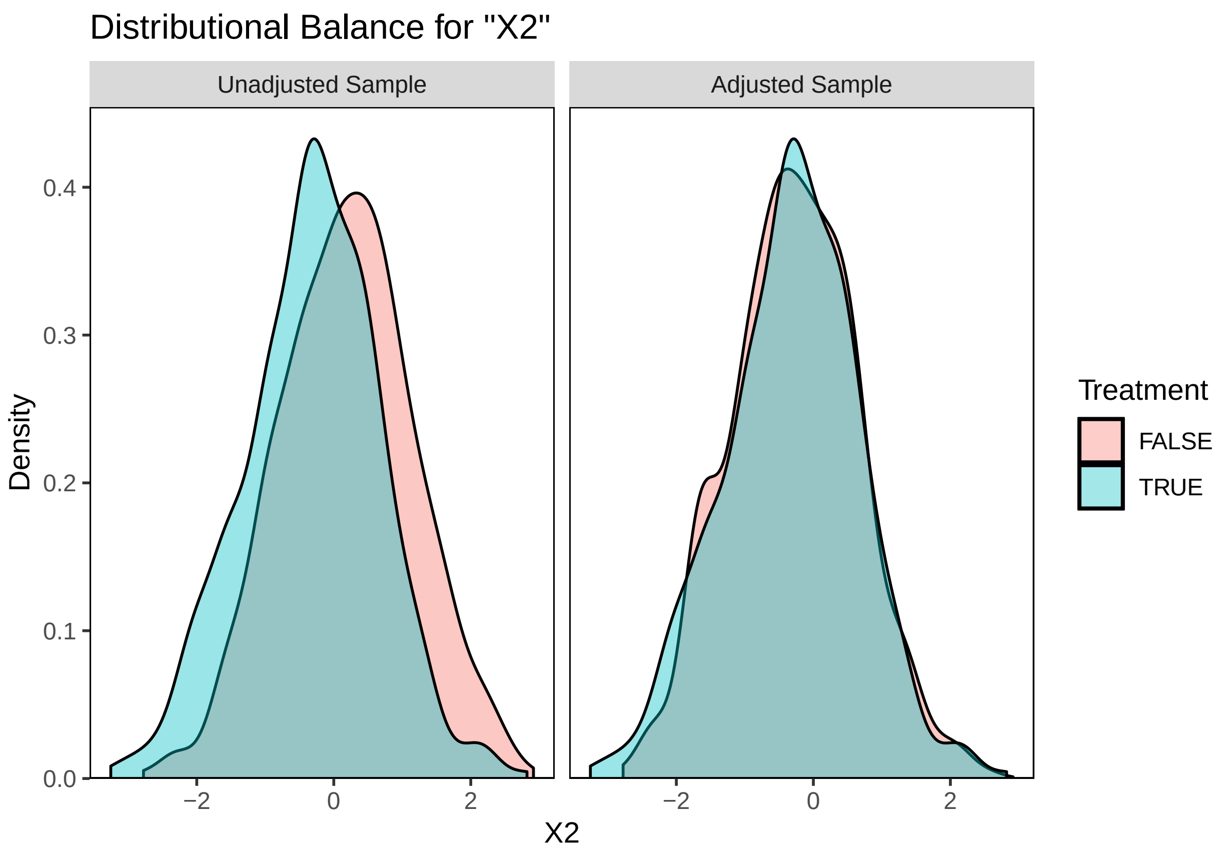
```
bal.plot(att.m1, which = "both", var.name = "X2", covs = match_vars, treat = (df$D==1))
```

```
## Warning: The following aesthetics were dropped during statistical transformation: weight
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



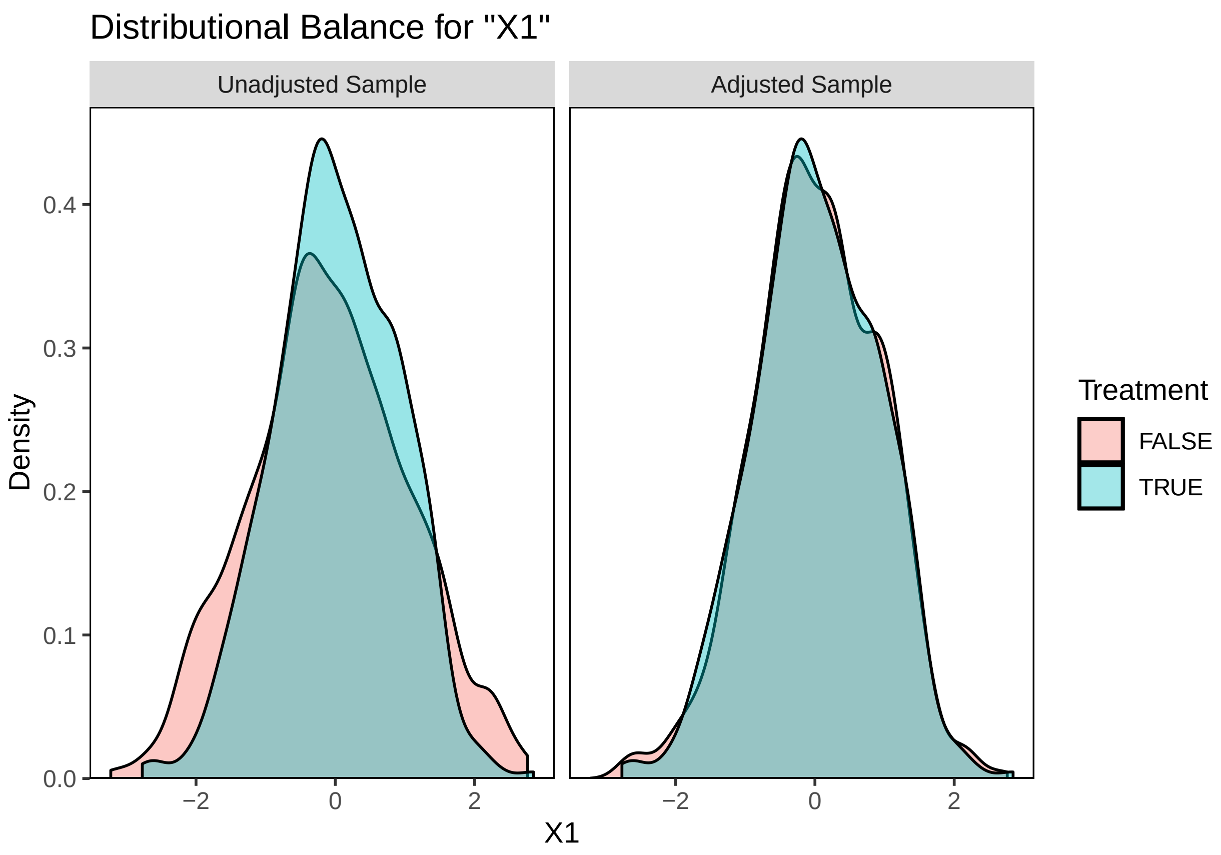
```r
bal.plot(att.m4, which = "both", var.name = "X2", covs = match_vars, treat = (df$D==1))
```

```
## Warning: The following aesthetics were dropped during statistical transformation: weight
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



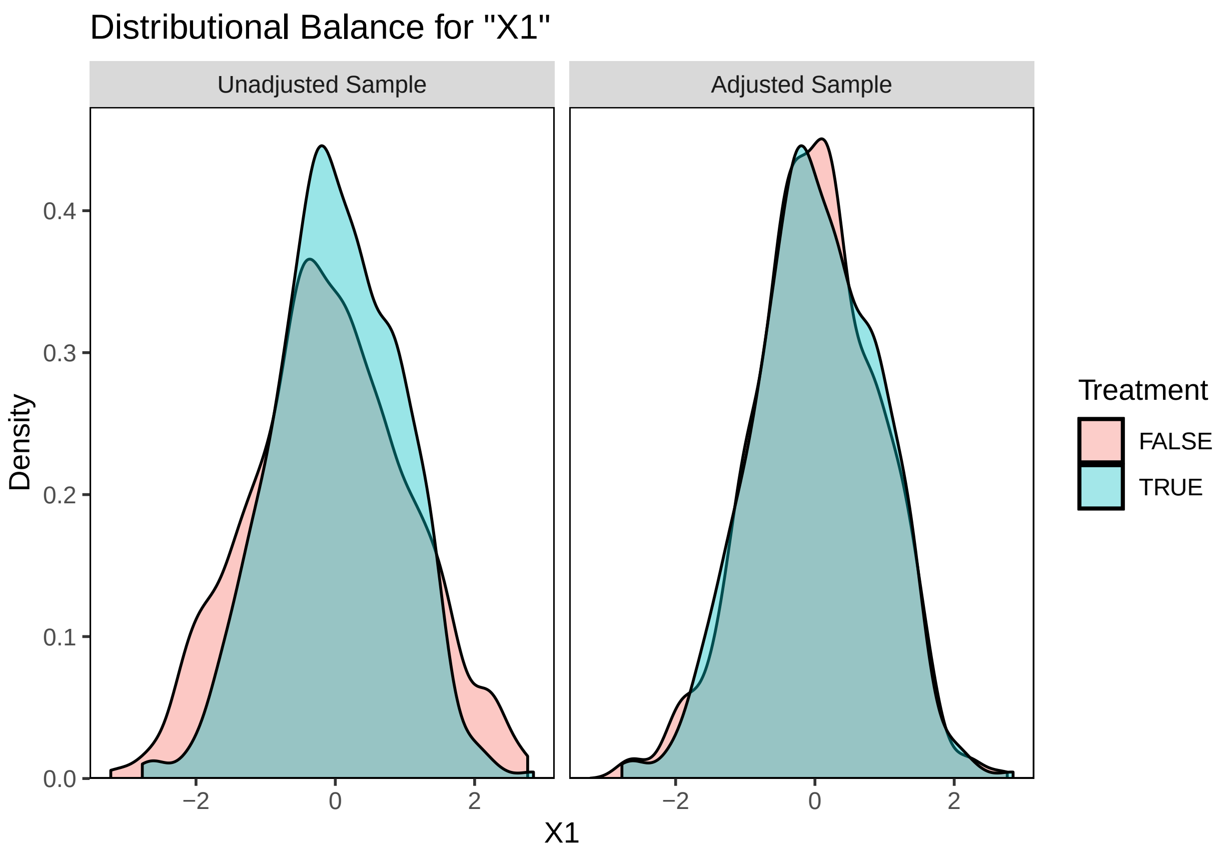
```
bal.plot(att.m1, which = "both", var.name = "X1", covs = match_vars, treat = (df$D==1))
```

```
## Warning: The following aesthetics were dropped during statistical transformation: weight
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



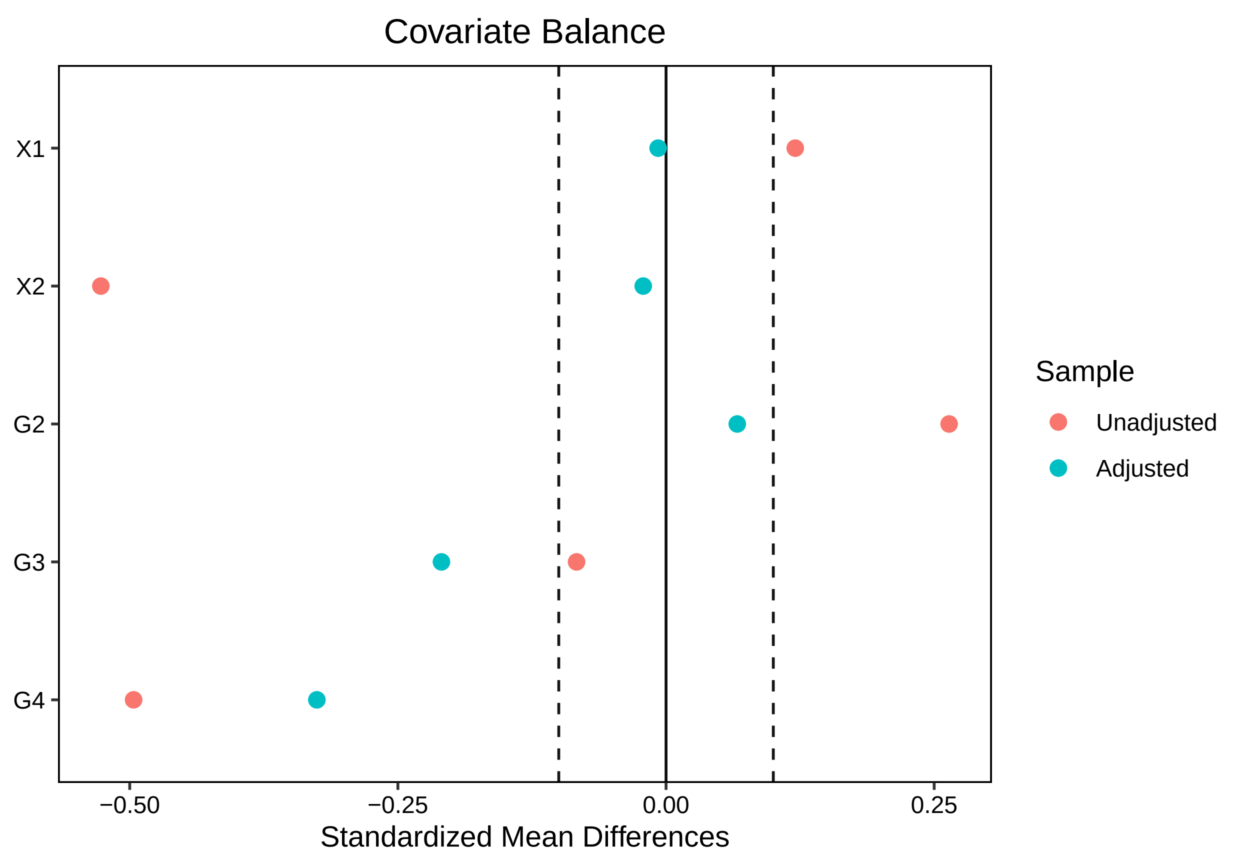
```
bal.plot(att.m4, which = "both", var.name = "X1", covs = match_vars, treat = (df$D==1))
```

```
## Warning: The following aesthetics were dropped during statistical transformation: weight
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```

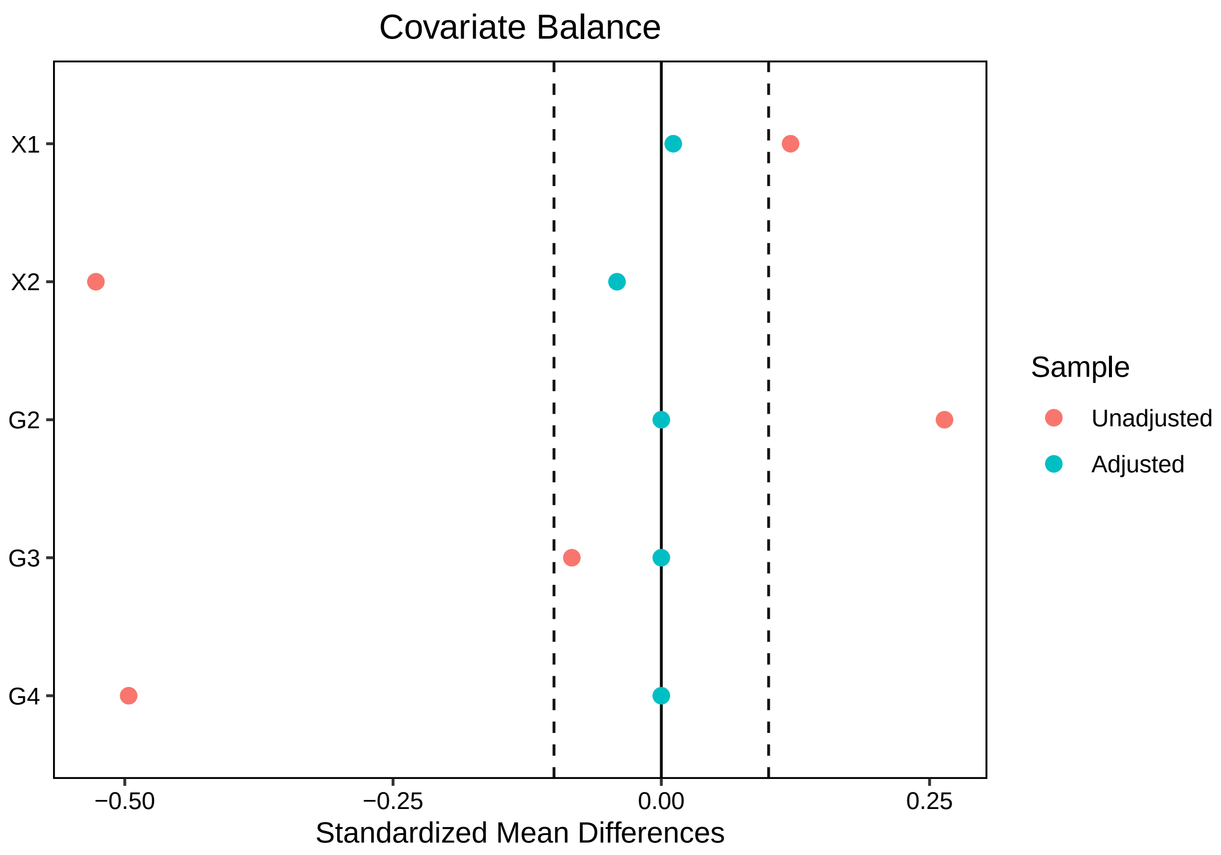


```
###

# 3. LOVE PLOTS
love.plot(att.m1, threshold = 0.1, binary = 'std',treat = (df$D==1), covs = match_vars)
```



```
love.plot(att.m4, threshold = 0.1, binary = 'std',treat = (df$D==1), covs = match_vars)
```


Covariate Balance
X1
X2
G2
G3
G4
−0.50
−0.25
0.00
0.25
Standardized Mean Differences
Sample
Unadjusted
Adjusted