PLSC 30600 - Lab 4

01/31/2025

Analyzing an observational study: Keriakes et. al. (2000)

- This lab looks at an observational study looking at the effect of an experimental drug on survival rates after percutaneous coronary interventions (PCI) - interventions to open blocked coronary arteries (via angioplasty + insertion of a stent).
- Keriakes et. al. (2000) look at the effect of a drug called abciximab, a type of antiplatlet drug used to prevent blood clots during these coronary artery interventions.
- The dataset consists of 996 patients under observation at Ohio Heart Health, Christ Hospital, Cincinatti in 1997. Each patient received a percutaneous coronary intervention and was under observation for at least 6 months.
- At the end of the 6 month period, survival was recorded. Some patients received treatment with abciximab but in a non-random manner. Patients who doctors believed had more severe cases of heart disease were more likely to receive the drug.

```
pci <- read_csv("pci.csv")
```

The relevant variables are

- `sixMonthSurvive` - Survival at 6 months - Main outcome of interest
- `abcix` - Treatment with abciximab - Main treatment of interest

Some of the other observed pre-treatment covariates are

- `stent` - Coronary stent deployment; numeric, with 1 meaning YES and 0 meaning NO.
- `female` - Female gender; numeric, with 1 meaning YES and 0 meaning NO.
- `diabetic` - Diabetes mellitus diagnosis; numeric, with 1 meaning YES and 0 meaning NO
- `acutemi` - Acute myocardial infarction within the previous 7 days; numeric, with 1 meaning YES and 0 meaning NO.

Balance checks before adjustment

Our naive estimate of the treatment effect without adjustment suggests a small positive effect on survival probability (about 3 percentage point).

```
lm_robust(sixMonthSurvive ~ abcix, data=pci)
```

```
##               Estimate Std. Error   t value  Pr(>|t|)   CI Lower   CI Upper  DF
## (Intercept) 0.94966443 0.01268657 74.855861 0.0000000 0.92476889 0.97455997 994
## abcix       0.03457626 0.01353525  2.554534 0.0107811 0.00801531 0.06113721 994
```

- However, we have reason to believe this is biased for the true ATE of abciximab treatment since it was administered to patients with more severe illness. Let's diagnose the covariate balance.
- The R package `cobalt` is really useful for generating balance tables for weighting estimators - it does a lot of the annoying pre-processing automatically (like standardizing the variables to have a standard deivation of 1).

```
# Subset the covariates we want to a data-frame
pci_covs <- pci %>% select(stent, female, diabetic, height, acutemi)
```

```r
# cobalt::bal.tab() will take a matrix of covariates and a treatment indicator and give standardized me
# it doesn't do t-tests by design (since they can be misleading and are discouraged by more recent work
# standardized mean differences above .1 are often a good threshold to be concerned (https://pubmed.ncb
# s.d.denom="pooled": Setting it to "pooled" means that the standard deviation used in the calculation
balance_tab <- bal.tab(pci_covs, treat=pci$abcix, s.d.denom="pooled")
balance_tab
```

```
## Balance Measures
##             Type Diff.Un
## stent     Binary  0.1210
## female    Binary -0.0550
## diabetic  Binary -0.0636
## height   Contin. -0.0003
## acutemi   Binary  0.1187
##
## Sample sizes
##     Control Treated
## All     298     698
```

```r
# We can look not only at the differences in means but in differences of higher-order moments
# The balance assessment will check not only the linear terms of the covariates but also their squared
balance_tab_sqd  <- bal.tab(pci_covs, treat=pci$abcix, s.d.denom="pooled", poly=2)
balance_tab_sqd
```

```
## Balance Measures
##             Type Diff.Un
## stent     Binary  0.1210
## female    Binary -0.0550
## diabetic  Binary -0.0636
## height   Contin. -0.0003
## acutemi   Binary  0.1187
## height^2  Contin.  0.0003
##
## Sample sizes
##     Control Treated
## All     298     698
```

Propensity score estimation

Now let's estimate the propensity score model. Let's try the easiest model with no interactions or polynomial terms in the linear predictor

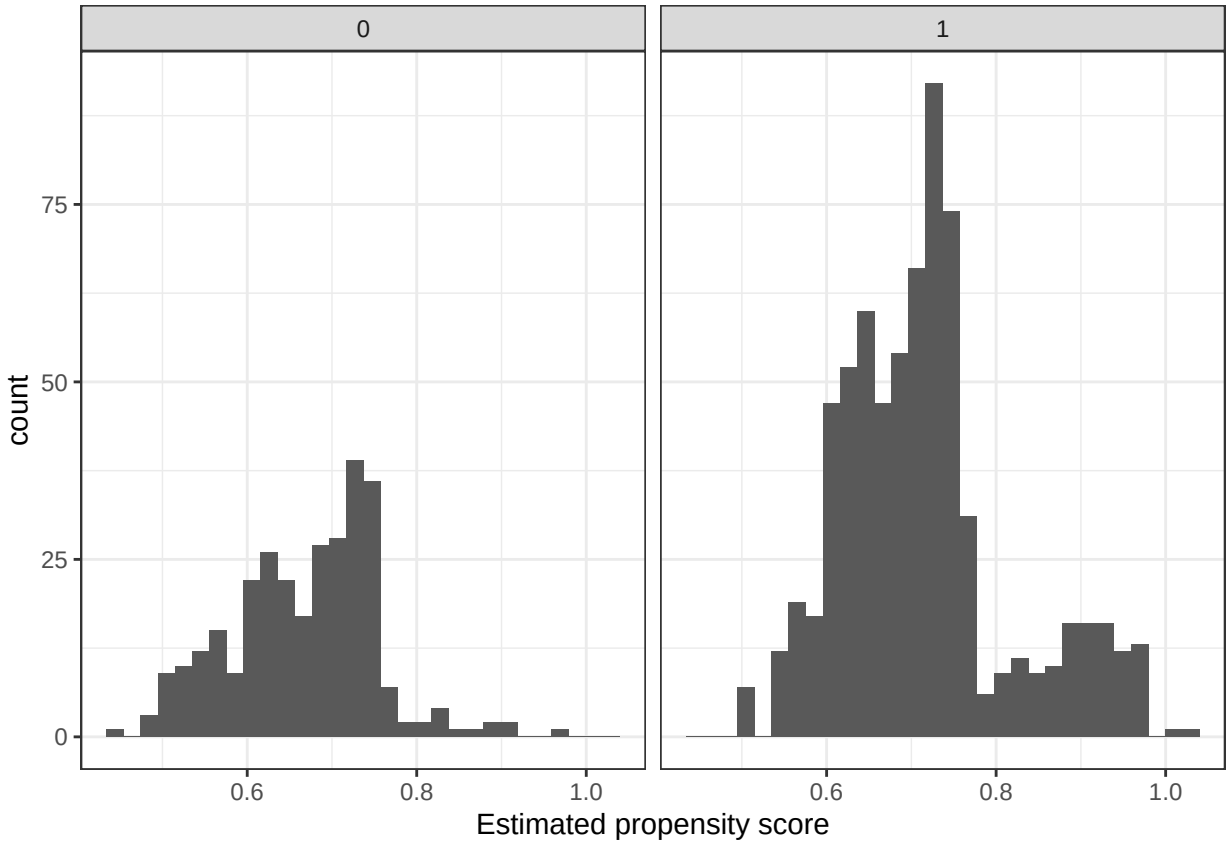
```r
# Fit a propensity score model
pscore_model <- lm_robust(abcix ~ stent + female + diabetic + height + acutemi,
                          data=pci)
summary(pscore_model)
```

```
##
## Call:
## lm_robust(formula = abcix ~ stent + female + diabetic + height +
##     acutemi, data = pci)
##
## Standard error type:  HC2
##
```

```
## Coefficients:
##              Estimate Std. Error t value  Pr(>|t|)  CI Lower   CI Upper  DF
## (Intercept)  1.117874   0.315958   3.538 4.218e-04  0.497849  1.7378989 990
## stent        0.109908   0.031112   3.533 4.305e-04  0.048854  0.1709612 990
## female      -0.086995   0.039764  -2.188 2.892e-02 -0.165027 -0.0089631 990
## diabetic    -0.065864   0.035056  -1.879 6.056e-02 -0.134657  0.0029283 990
## height      -0.002769   0.001778  -1.557 1.197e-01 -0.006259  0.0007204 990
## acutemi      0.203122   0.031374   6.474 1.497e-10  0.141555  0.2646886 990
##
## Multiple R-squared:  0.04638 ,   Adjusted R-squared:  0.04157
## F-statistic: 14.49 on 5 and 990 DF,  p-value: 1.004e-13
```

```
# Predict the propensity score
pci$e <- predict(pscore_model, newdata = pci, type = "response")
# type = "response" gives us the probabilities

# Let's see the histogram of the propensity scores among treated and control
pci %>% ggplot(aes(x=e)) +
  geom_histogram(bins=30) +
  facet_wrap(~abcix) +
  xlab("Estimated propensity score") +
  theme_bw()
```



This is really quite good in terms of overlap (if we got the model right). Not a lot of 0 or 1 propensity scores. Now let's make the Inverse Probability Treatment Weighting (IPTW) weights

```
# Generate the weights
pci$wt[pci$abcix == 1] <- 1/pci$e[pci$abcix==1]
```

```
## Warning: Unknown or uninitialised column: `wt`.
```

```
pci$wt[pci$abcix == 0] <- 1/(1 - pci$e[pci$abcix==0])


# Generate a point estimate
iptw_est <- lm_robust(sixMonthSurvive ~ abcix, data=pci, weights=wt)
point_wtd <- coef(iptw_est)[2]
point_wtd
```

```
##      abcix
## 0.04326576
```

Recall that our unadjusted estimate was around 3 percentage points. After adjustment, the estimated effect is closer to 4 percentage points. This is in line with our expectations that the selection-into-treatment bias negatively biased the naive difference-in-means since patients with worse conditions were more likely to get treatment.

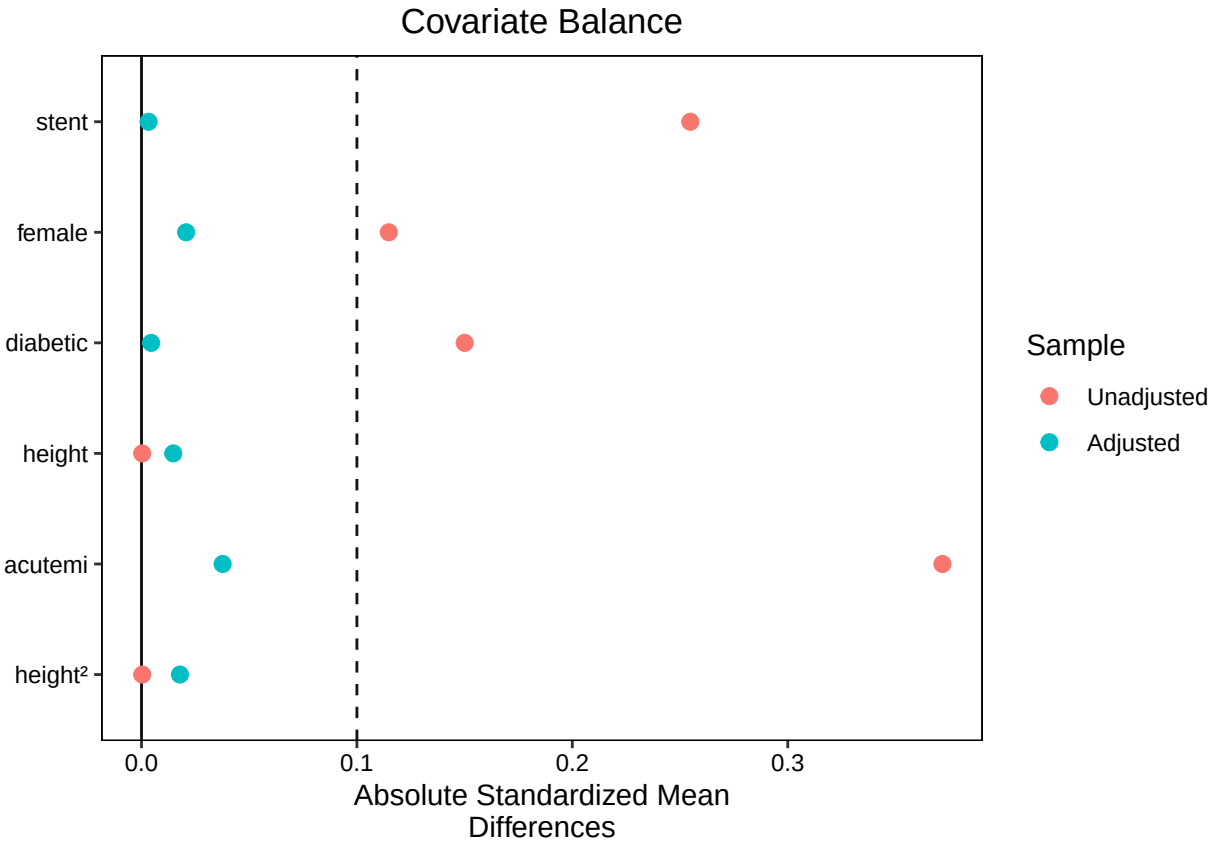
Balance checks after adjustment

We'll come back to inference in a second. But first, let's see how well covariate balance has improved with the weights.

```
# Pass the IPTW weights
balance_tab_wt <- bal.tab(pci_covs, treat=pci$abcix, s.d.denom="pooled", weights=pci$wt)
balance_tab_wt
```

```
## Balance Measures
##             Type Diff.Adj
## stent     Binary   0.0016
## female    Binary   0.0099
## diabetic  Binary  -0.0019
## height   Contin.  -0.0147
## acutemi   Binary   0.0120
##
## Effective sample sizes
##            Control Treated
## Unadjusted   298.    698.
## Adjusted     236.4   685.7
```

```
# A mega useful visualization tool to compare unadjusted vs. adjusted
# is the "love" plot (named after biostatistician Thomas Love)
iptw_love_plot <- love.plot(pci_covs, treat=pci$abcix, s.d.denom="pooled", abs=T, poly=2,
                            binary = "std", weights=pci$wt, thresholds= c(m=.1))
iptw_love_plot
```



Note that we even get improvement on the higher-order moments of some of the continuous covariates even though we didn't include them in the model. However, we don't get improvement *everywhere* - post-weighting balance is slightly worse for height (on which there was basically no imbalance to begin with). But overall, weighting does a pretty good job of attenuating the difference in the *observed* covariates between treated and control. Of course, you should keep in mind that this is only addressing balance on covariates that we *observe* - if there's unobserved confounding (which we're assuming there's not), we could be making it worse!

Bootstrapping

Recall again that bootstrapping is a way of approximating the sampling distribution of an estimator and estimating features of it (such as the variance), by resampling from our sample. With independent observations, the nonparametric bootstrap repeatedly resamples observations *with replacement* from the sample and computes an estimate for each resample.

```
set.seed(60637)
nBoot <- 1000 # Number of iterations
ate_boot <- rep(NA, nBoot) # Placeholder to store estimates

# For each iteration
for(boot in 1:nBoot){

  # Resample rows with replacement
  pci_boot <- pci[sample(1:nrow(pci), nrow(pci), replace=T),] #replace = T is key!

  # Fit the propensity score model on the bootstrapped data
  pscore_model <- lm_robust(abcix ~ stent + female + diabetic + height + acutemi,
                      data=pci)
```

```
  # Get the propensity scores for each observation
  pci$e <- predict(pscore_model, newdata = pci)

  # Calculate the weights
  pci_boot$wt_boot <- NA
  pci_boot$wt_boot[pci_boot$abcix == 1] <- 1/pci_boot$e[pci_boot$abcix==1]
  pci_boot$wt_boot[pci_boot$abcix == 0] <- 1/(1 - pci_boot$e[pci_boot$abcix==0])

  # weighted difference-in-means
  boot_reg <- lm_robust(sixMonthSurvive ~ abcix, data=pci_boot, weights=wt_boot)

  # Store the weighted difference-in-means
  ate_boot[boot] <- coef(boot_reg)[2]

}

# Take the SD of the ate_boot to get our estimated SE - can do asymptotic inference
sd(ate_boot)
```

```
## [1] 0.01822421
```

```
# Asymptotic 95\% CI
c(point_wtd - qnorm(.975)*sd(ate_boot),
  point_wtd + qnorm(.975)*sd(ate_boot))
```

```
##       abcix       abcix
## 0.007546973 0.078984550
```

```
# Can also take quantiles to get CIs directly from the bootstrapped distribution (esp. if skewed)
quantile(ate_boot, c(.025, .975))
```

```
##       2.5%      97.5%
## 0.01103982 0.08531369
```

Our 95% confidence interval does not include 0 – we'd reject the null of no ATE at $\alpha = .05$.

Simulation Illustration:

We simulate a dataset with $n = 1000$ subjects.

- Two binary covariates: X_1 and X_2.
- A binary treatment variable T that follows a Bernoulli distribution with probability $0.3 + 0.2 \times X_1 + 0.45 \times X_2$. This introduces an imbalance in the treatment assignment, as the probability of receiving the treatment depends on the covariates.
- an outcome variable Y is generated for each subject, following a normal distribution with mean $T \times 2 + 0.2 \times X1 + 0.8 \times X2$. This means the outcome depends on both the treatment and the covariates.

```
# Simulate data
set.seed(60637)
n <- 1000 # number of subjects
X1 <- rbinom(n, 1, 0.5) # binary covariate
X2 <- rbinom(n, 1, 0.2) # binary covariate
T <- rbinom(n, 1, 0.3 + 0.2 * X1 + 0.45 * X2 ) # treatment assignment with imbalance

# outcomes depend on both treatment and covariates
```

```
Y <- rnorm(n, mean = T * 2+0.2*X1+0.8*X2)

data = tibble(Y, T, X1, X2)
```

Naive estimate of the treatment effect without adjustment:

```
estimatr::tidy(lm_robust(Y ~ T, data=data))
```

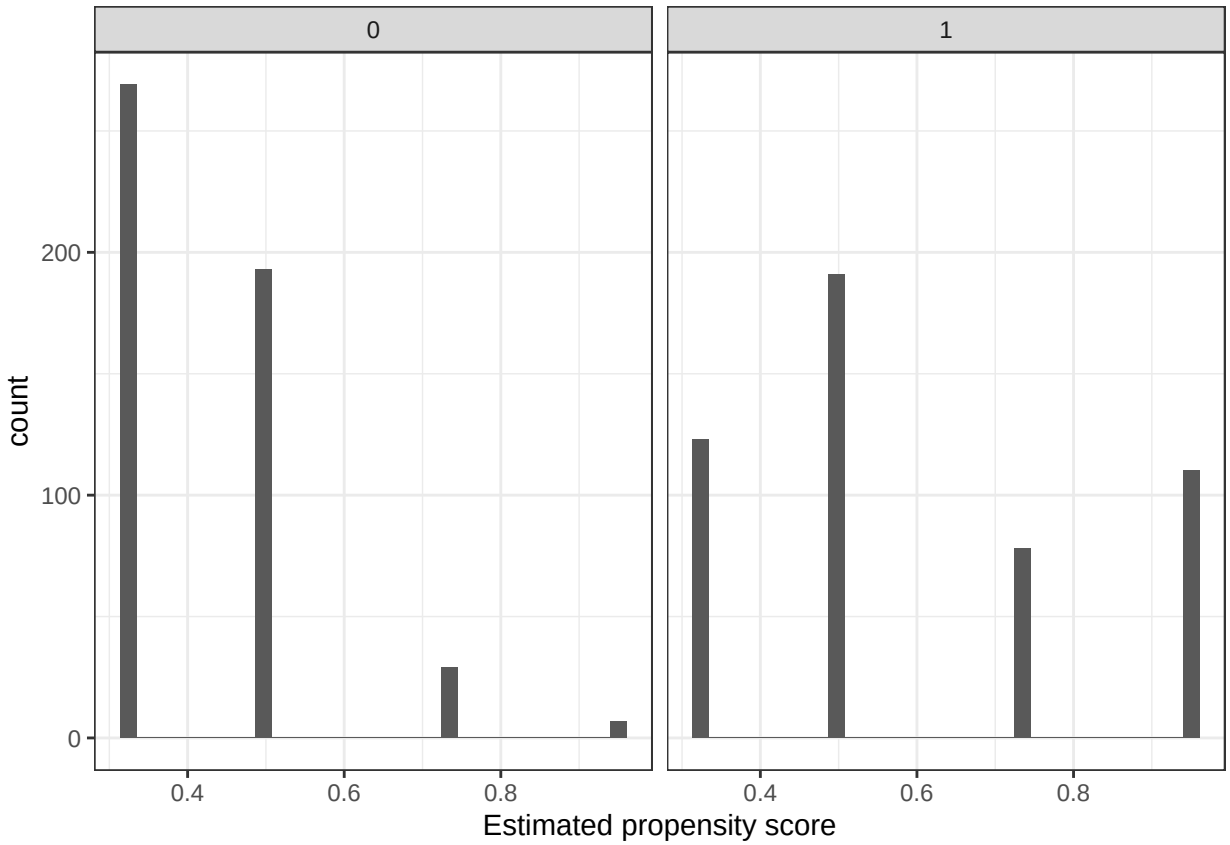
```
##          term  estimate  std.error statistic       p.value  conf.low conf.high
## 1 (Intercept) 0.1002126 0.04493901  2.229969  2.597129e-02 0.0120268 0.1883984
## 2           T 2.3539668 0.06603874 35.645241 3.735406e-180 2.2243761 2.4835575
##    df outcome
## 1 998       Y
## 2 998       Y
```

We can see that the CI does not cover the true average treatment effect of 2.

Propensity score estimation

```
ps_model <- lm_robust(T ~ X1*X2, data = data)
data$ps <- predict(ps_model, newdata = data)


# Let's see the histogram of the propensity scores among treated and control
data %>% ggplot(aes(x=ps)) +
  geom_histogram(bins=30) +
  facet_wrap(~T) +
  xlab("Estimated propensity score") +
  theme_bw()
```



Propensity score weighting

```
# Apply IPW
data$weights <- ifelse(T == 1, 1/data$ps, 1/(1-data$ps))

# Assess balance

bal_table_unwtd <- bal.tab(T ~ X1 + X2, data = data, binary = "std", s.d.denom="pooled",
                      )
print(bal_table_unwtd)
```

```
## Balance Measures
##      Type Diff.Un
## X1 Binary  0.4040
## X2 Binary  0.7786
##
## Sample sizes
##     Control Treated
## All     498     502
```

```
# cobalt::bal.tab() will take a matrix of covariates and a treatment indicator and give standardized me
```

```
bal_table_wtd <- bal.tab(T ~ X1 + X2, data = data, binary = "std", weights = data$weights,
                      method = "weighting")
```

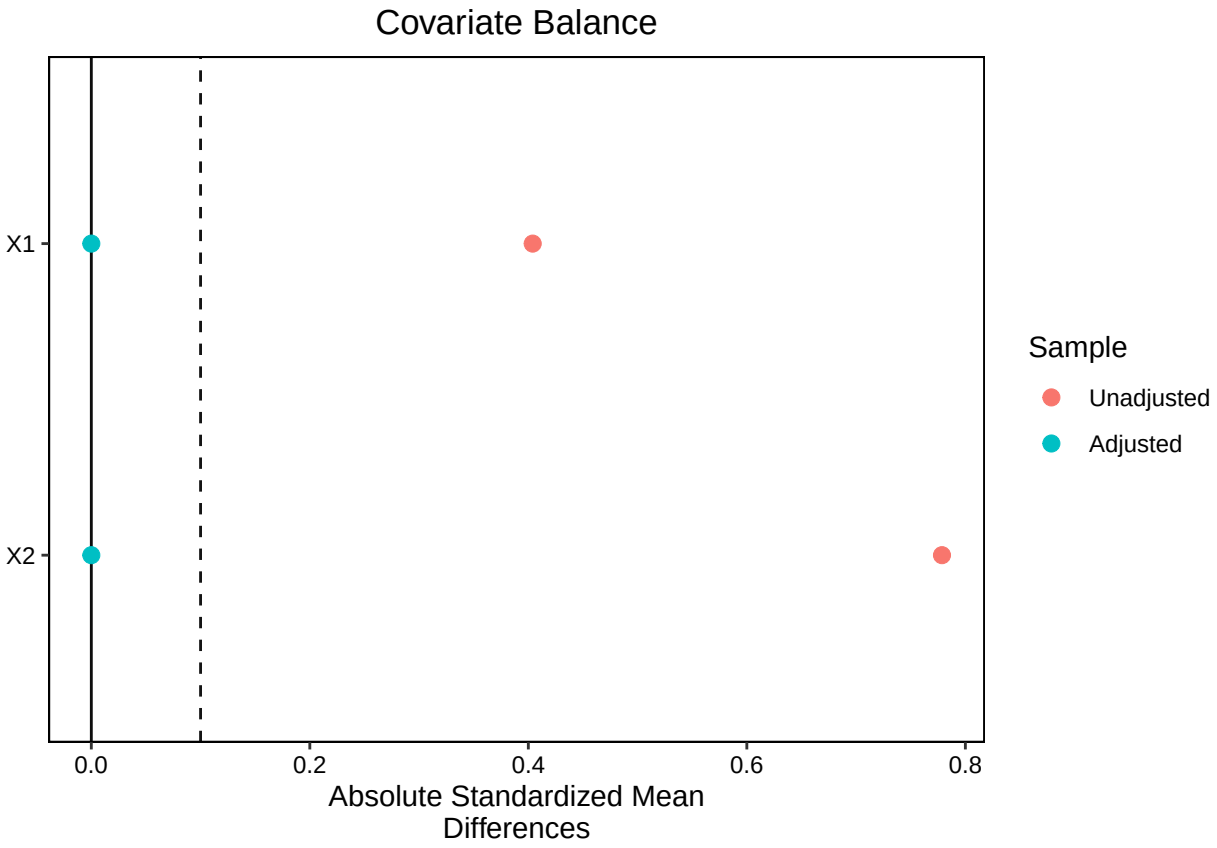
```
## Note: `s.d.denom` not specified; assuming "pooled".
```

```
print(bal_table_wtd)
```

```
## Balance Measures
##      Type Diff.Adj
## X1 Binary        0
## X2 Binary       -0
##
## Effective sample sizes
##            Control Treated
## Unadjusted  498.     502.
## Adjusted    271.32   436.2
```

Then, we can see that the balance is improved after weighting.

```
# A mega useful visualization tool to compare unadjusted vs. adjusted
# is the "love" plot (named after biostatistician Thomas Love)
iptw_love_plot <- cobalt::love.plot(data %>% select(X1, X2), treat=data$T, s.d.denom="pooled", abs=TRUE
                                    binary = "std", weights=data$weights, thresholds= c(m=.1))
iptw_love_plot
```



Next, we estimate the treatment effect using IPW.

```
# Generate a point estimate
iptw_est <- lm_robust(Y ~ T, data=data, weights=weights)
point_wtd <- coef(iptw_est)[2]
point_wtd
```

```
##        T
## 2.089427
```

How do we get the standard error? We can use the bootstrap.

Bootstrapping

Recall again that bootstrapping is a way of approximating the sampling distribution of an estimator and estimating features of it (such as the variance), by resampling from our sample. With independent observations, the nonparametric bootstrap repeatedly resamples observations *with replacement* from the sample and computes an estimate for each resample.

```
set.seed(60637)
nBoot <- 1000 # Number of iterations
ate_boot <- rep(NA, nBoot) # Placeholder to store estimates

# For each iteration
for(boot in 1:nBoot){

  # Resample rows with replacement
  data_boot <- data[sample(1:nrow(data), nrow(data), replace=TRUE),] #replace = T is key!

  # Fit the propensity score model on the bootstrapped data
  pscore_model <- lm_robust(T ~ X1*X2,
                      data=data_boot)

  # Get the propensity scores for each observation
  data_boot$ps <- predict(pscore_model, newdata = data_boot)

  # Calculate the weights
  data_boot$wt_boot <- NA
  data_boot$wt_boot <- ifelse(T == 1, 1/data_boot$ps, 1/(1-data_boot$ps))

  # weighted difference-in-means
  boot_reg <- lm_robust(Y ~ T, data=data_boot, weights=data_boot$wt_boot)

  # Store the weighted difference-in-means
  ate_boot[boot] <- coef(boot_reg)[2]

}

# Take the SD of the ate_boot to get our estimated SE - can do asymptotic inference
sd(ate_boot)
```

```
## [1] 0.1069011
```

```
# Asymptotic 95\% CI
c(point_wtd - qnorm(.975)*sd(ate_boot),
  point_wtd + qnorm(.975)*sd(ate_boot))
```

```
##        T        T
## 1.879904 2.298949
```