Propensity Score Analysis in R - Basics

Statistical Horizons

September 18-20, 2025

Helpful Resources

- Propensity Score Analysis support site: https://ssw.unc.edu/psa/
- MatchIt package website: https://kosukeimai.github.io/MatchIt/
- Cobalt package website: https://ngreifer.github.io/cobalt/
- List of R software for propensity score analysis: https://www.elizabethstuart.org/psoftware/
- Stuart, E. A. (2010). Matching Methods for Causal Inference: A Review and a Look Forward. *Statistical Science*, *25*(1), 1–21. https://doi.org/10.1214/09-STS313
- Greifer, N., & Stuart, E. A. (2023). Choosing the Causal Estimand for Propensity Score Analysis of Observational Studies (arXiv:2106.10577). *arXiv*. https://doi.org/10.48550/arXiv.2106.10577

Conceptual Review

Basic Concepts

Causal inference is inference about *counterfactual outcomes* or *potential outcomes*. An example of a counterfactual is: What would have happened to the treated subjects, had they not received treatment?

(*Notation:* Let w represent a dichotomous treatment variable, where 1 = treated and 0 = untreated, let Y represent a dichotomous outcome variable. Y_{1i} is the potential outcome for unit i with treatment, and Y_{0i} is the potential outcome for unit i without treatment.)

An **individual causal effect** is given by:

$$\tau_i = Y_{1i} - Y_{0i}$$

The **Neyman-Rubin counterfactual framework** states that individuals selected into treatment and non-treatment groups have potential outcomes in both states: the one in which they are observed and the one in which they are not observed, which can be expressed as:

$$Y_i = W_i Y_{1i} + (1 - W_i) Y_{0i}$$

This framework assumes the **Stable Unit Treatment Value Assumption (SUTVA)**, which assumes that (1) potential outcomes for an individual must not be affected by treatment for other individuals (e.g., spill-over effect); and (2) there are no different versions of the treatment condition (e.g., individuals receive different or variable levels of treatment).

Because "the fundamental problem of causal inference" is that individual causal effects are unobservable, we generally focus on **average causal effects**. One causal estimand is the **average treatment effect (ATE)**, given by:

$$\tau_{ATE} = \frac{1}{N}\sum_{i=1}^{N}\{Y_{1i} - Y_{0i}\} \text{ or } \tau_{ATE} = \mathbb{E}[Y_{1i} - Y_{0i}]$$

If we let N_1 equal the number of treated units, then the **average treatment effect on the treated (ATT)** is given by:

$$\tau_{ATT} = \frac{1}{N_1}\sum_{i=1}^{N} W_i\{Y_{1i} - Y_{0i}\} \text{ or } \tau_{ATT} = \mathbb{E}[Y_{1i} - Y_{0i} | W_i = 1]$$

The **propensity score** is the probability of receiving the treatment given a vector of observed covariates, X_i:

$$\pi(X_i) \equiv \Pr(W_i = 1 | X_i)$$

Greedy Matching

Greedy nearest neighbor matching forms matches by pairing a control participant with a treated participant, if the absolute difference of their propensity scores is the smallest among all possible pairs of propensity scores (i.e., they are similar on their covariates). If a caliper is used, then a match is made only if the absolute difference is less than the caliper.

Strengths

- Allows various kinds of post-matching analysis
- Particularly useful if the outcome variable is non-normal, non-continuous (e.g., categorical dependent variable, time-to-event data)
- Mahalanobis metric matching can be used to find well-matched pairs when the number of units is small

Limitations

- Optimization is only done locally
- "Bias due to incomplete matching" when treated subjects are excluded[1]
- Requires a sizeable common support region

Optimal Matching

Optimal matching minimizes a global distance measure to form matches. This circumvents the problem in greedy matching, where the order in which the treated are matched affects the quality of the matches. According to Gu and Rosenbaum (1993), "optimal matching picks about the same controls [as greedy matching] but does a better job of assigning them to treated units." Thus, optimal matching and greedy matching tend to produce similar results, but optimal matching may produce better matched pairs.[2]

In optimal pair matching, typically a treated participant is matched to a single control. With variable ratio or variable matching, each treated participant matches to a variable number of controls (because some individuals may have more matches than others).

Strengths

- Finds well-matched pairs
- Full matching uses the entire sample
- Robust against violations of overlap in common support region

Limitations

- Does not necessarily produce better balanced groups than greedy matching
- Variable matching may increase bias due to poor matches

Matching with or without Replacement

According to Stuart (2010), matching with replacement is "particularly helpful in settings where there are few control individuals comparable to the treated individuals (e.g., Dehejia and Wahba, 1999). Additionally, when matching with replacement the order in which the treated individuals are matched does not matter. However, inference becomes more complex when matching with replacement, because the matched controls are no longer independent—-some are in the matched sample more than once and this needs to be accounted for in the outcome analysis, for example by using frequency weights. When matching with replacement it is also possible that the treatment effect estimate will be based on just a small number of controls; the number of times each control is matched should be monitored."[3]

A simulation study comparing 12 propensity score algorithms found that "matching with replacement did not result in estimates with less bias compared with the best-performing methods based on caliper matching

[1]Austin, P. C. (2014). A comparison of 12 algorithms for matching on the propensity score. *Statistics in Medicine*, *33*(6), 1057–1069. https://doi.org/10.1002/sim.6004

[2]Stuart, E. A. (2010). Matching Methods for Causal Inference: A Review and a Look Forward. *Statistical Science*, *25*(1), 1–21. https://doi.org/10.1214/09-STS313

[3]Stuart, E. A. (2010). Matching Methods for Causal Inference: A Review and a Look Forward. *Statistical Science*, *25*(1), 1–21. https://doi.org/10.1214/09-STS313

without replacement. Furthermore, matching with replacement resulted in estimates that displayed greater variability and that had higher MSE compared with estimates obtained using caliper matching without replacement."[4]

According to Abadie & Imbens (2006), "Matching with replacement produces matches of higher quality than matching without replacement by increasing the set of possible matches. In addition, matching with replacement has the advantage that it allows us to consider estimators that match all units, treated as well as controls, so that the estimand is identical to the population average treatment effect."[5]

Propensity Score Weighting

Unlike matching, propensity score weighting balances data by using propensity scores to create weights. It makes the estimate of the sample average treatment effect or its inference to the population average treatment effect a weighted average of the difference between observed and potential outcomes. It is similar to the weighted analysis that is conducted when analyzing complex survey designs. To estimate the ATE, the treatment weights are $1/\hat{e}(x)$, where $\hat{e}(x)$ is the propensity score, and the control weights are $1/(1-\hat{e}(x))$—this is known as the inverse probability of treatment weights (IPTW) estimator. To estimate the ATT, the treatment weight is 1, and the control weights are $\hat{e}(x)/(1-\hat{e}(x))$.

Strengths

- Permits most types of multivariate outcome analyses
- Does not require an outcome variable that is continuous or normally distributed
- Retains most participants in the outcome analysis
- Flexible in the context of complex surveys[6]

Limitations

- Variance can be large if weights are extreme[7]

Matching Estimators

Matching estimators directly impute missing potential outcomes at the unit level using a vector norm. After imputing the missing data, matching estimators can be used to estimate various average treatment effects. Instead of using logistic regression to predict propensity scores, matching estimators use a vector norm to calculate distances on the observed covariates between a treated case and each of its potential control cases. These calculations use either the inverse of the sample variance matrix or the inverse of the sample variance-covariance matrix. When using the inverse of the sample variance-covariance matrix, the matching estimator calculates Mahalanobis metric distances.

Strengths

- Calculate a variety of average treatment effects
- Estimate effects for both the sample and the population

[4] Austin, P. C. (2014). A comparison of 12 algorithms for matching on the propensity score. *Statistics in Medicine*, *33*(6), 1057–1069. https://doi.org/10.1002/sim.6004

[5] Abadie, A., & Imbens, G. W. (2006). Large Sample Properties of Matching Estimators for Average Treatment Effects. *Econometrica*, *74*(1), 235–267. https://doi.org/10.1111/j.1468-0262.2006.00655.x

[6] DuGoff, E. H., Schuler, M., & Stuart, E. A. (2014). Generalizing Observational Study Results: Applying Propensity Score Methods to Complex Surveys. *Health Services Research*, *49*(1), 284–303. https://doi.org/10.1111/1475-6773.12090

[7] Austin, P. C., & Stuart, E. A. (2015). Moving towards best practice when using inverse probability of treatment weighting (IPTW) using the propensity score to estimate causal treatment effects in observational studies. *Statistics in Medicine*, *34*(28), 3661–3679. https://doi.org/10.1002/sim.6607

Limitations

- Require a sizeable common support region
- More sensitive to the violation of the strongly ignorable assumption

3.5 Computer Lab: Running Greedy Matching and GBR with R

Greedy Nearest Neighbor Matching

Greedy nearest neighbor matching forms matches by pairing a control participant with a treated participant, if the absolute difference of their propensity scores is the smallest among all possible pairs of propensity scores (i.e., they are similar on their covariates). If a caliper is used, then a match is made only if the absolute difference is less than the caliper.

Load Packages

The `haven` and `sjlabelled` packages are used to load and clean Stata data files (.dta); the `MatchIt` package contains functions for greedy matching[8]; the `cobalt` package contains functions for balance checking; and the `tidyverse` package is loaded for its data manipulation functions.

```r
if (!require("pacman")) install.packages("pacman")
pacman::p_load(
  haven, sjlabelled, cobalt, MatchIt, tidyverse, kableExtra, survival, psych, here
)
```

Description of Data

This data is a sample of 2,758 children from the National Survey of Child and Adolescent Well-Being (NSCAW), a nationally representative, longitudinal survey of children and families who have been the subjects of investigation by Child Protective Services. Two waves of NSCAW data were used: baseline information between October 1999 and December 2000 and the 18-months follow-up. The sample was limited to children who lived at home (e.g., were not in foster care) and whose primary caregivers were female (because the vast majority of primary caregivers in NSCAW were females). The treatment condition is `aodserv` or caregivers who received (aodserv = 1) or did not receive (aodserv = 0) substance abuse services. Two matching procedures are illustrated here. In Section 5.8.1 of the PSA-R code, 12 matching schemes are shown.[9]

Load and Clean Data

```r
# Load Data
gm_df0 <- haven::read_dta(here("data", "chpt5_1_original.dta"))

# Inspect Data
str(gm_df0)

# Remove Stata Labels and Formats
gm_df0 <- gm_df0 %>%
  haven::zap_formats() %>%
  sjlabelled::remove_all_labels() %>%
  as_tibble()

# Inspect Data
str(gm_df0)
```

[8]https://kosukeimai.github.io/MatchIt/articles/matching-methods.html
[9]https://ssw.unc.edu/psa/

```r
psych::describe(gm_df0)
nrow(gm_df0)
table(gm_df0$aodserv)
```

Sort Data

When non-treated cases have the same propensity score values, their matches will depend on the order of the data. Therefore, it's important to order the observations randomly.

```r
set.seed(1000)
gm_df <- gm_df0 %>%
  add_column(runif = runif(nrow(.)), .before = "PSH17A") %T>% print() %>%
  arrange(runif) %T>% print() %>%
  select(-runif)
```

Check Balance Before Matching

Because all of the variables that will be used for predicting the propensity scores of service receipt are categorical, we can use `chisq.test` to check their balance before matching. Before we do that, we convert these categorical variables from numeric variables to factor variables using `as.factor()`. This will help functions such as `bal.tab()` to automatically calculate appropriate balance statistics.

```r
# Change categorical independent variables to factor variables, except the id and
# and aodserv treatment variables
gm_df <- gm_df %>%
  mutate(across(c(-id, -aodserv), as.factor))

# Marital Status (large sample size, therefore continuity correction not needed)
chisq.test(gm_df$aodserv, gm_df$married, correct = F)
```

```
##
##  Pearson's Chi-squared test
##
## data:  gm_df$aodserv and gm_df$married
## X-squared = 2.9705, df = 1, p-value = 0.0848
```

```r
# All Variables
gm_df %>%
  select(
    married, educ, pov, employ, open, race, chdage, cgage, CRA47A, mental,
    arrest, PSH17A, maltx, ra, cidi, cgneed, cwwrep, aodserv
  ) %>%
  pivot_longer(-aodserv, names_to = "variable") %>%
  group_by(variable) %>%
  nest() %>%
  mutate(bivariate.test = map(data, ~chisq.test(.$aodserv, .$value, correct = F))) %>%
  mutate(statistic = map(bivariate.test, ~ round(.$statistic, 3))) %>%
  mutate(p.value = map(bivariate.test, ~ round(.$p.value, 3))) %>%
  unnest(cols = c(statistic, p.value)) %>%
  select(variable, statistic, p.value)
```

```
## # A tibble: 17 x 3
## # Groups:   variable [17]
##    variable statistic p.value
##    <chr>        <dbl>   <dbl>
##  1 married       2.97   0.085
##  2 educ         10.5    0.005
##  3 pov          11.3    0.023
##  4 employ       23.1    0
##  5 open         58.4    0
##  6 race         11.5    0.009
##  7 chdage       55.4    0
##  8 cgage         3.56   0.313
##  9 CRA47A       17.5    0
## 10 mental       92.5    0
## 11 arrest      127.     0
## 12 PSH17A      179.     0
## 13 maltx        49.7    0
## 14 ra          585.     0
## 15 cidi        157.     0
## 16 cgneed      139.     0
## 17 cwwrep     1240.     0
```

```
# Fisher's Exact Test for cgage
(c1 <- chisq.test(gm_df$aodserv, gm_df$cgage, correct = F))
```

```
##
##  Pearson's Chi-squared test
##
## data:  gm_df$aodserv and gm_df$cgage
## X-squared = 3.5619, df = 3, p-value = 0.3128
```

```
c1$expected
```

```
##               gm_df$cgage
## gm_df$aodserv        0         1        2         3
##             0 38.35388 1698.2741 582.4438 140.92821
##             1  4.64612  205.7259  70.5562  17.07179
```

```
fisher.test(gm_df$aodserv, gm_df$cgage)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  gm_df$aodserv and gm_df$cgage
## p-value = 0.288
## alternative hypothesis: two.sided
```

Alternatively, the `cobalt` package provides several convenient functions for assessing balance.

The standardized mean difference (SMD) (also referred to as the "normalized difference") is a commonly used balance measure.[10] It is calculated as the difference in means of a covariate across the treatment groups,

[10]The disadvantage of hypothesis tests is "they are influenced by sample size, which fluctuates during adjustment, and the theory behind them is inappropriate because balance is a quality solely of the sample in question, not in relation to a population" (https://cran.r-project.org/web/packages/cobalt/vignettes/cobalt.html).

divided by the standard deviation in the treated group (ATT), the control group (ATC), or the pooled standard deviation (ATE). See `?cobalt::col_w_smd` for additional options. Stuart et al. (2013) recommend 0.1 or 0.25 as reasonable cut-offs for acceptable standardized biases.[11]

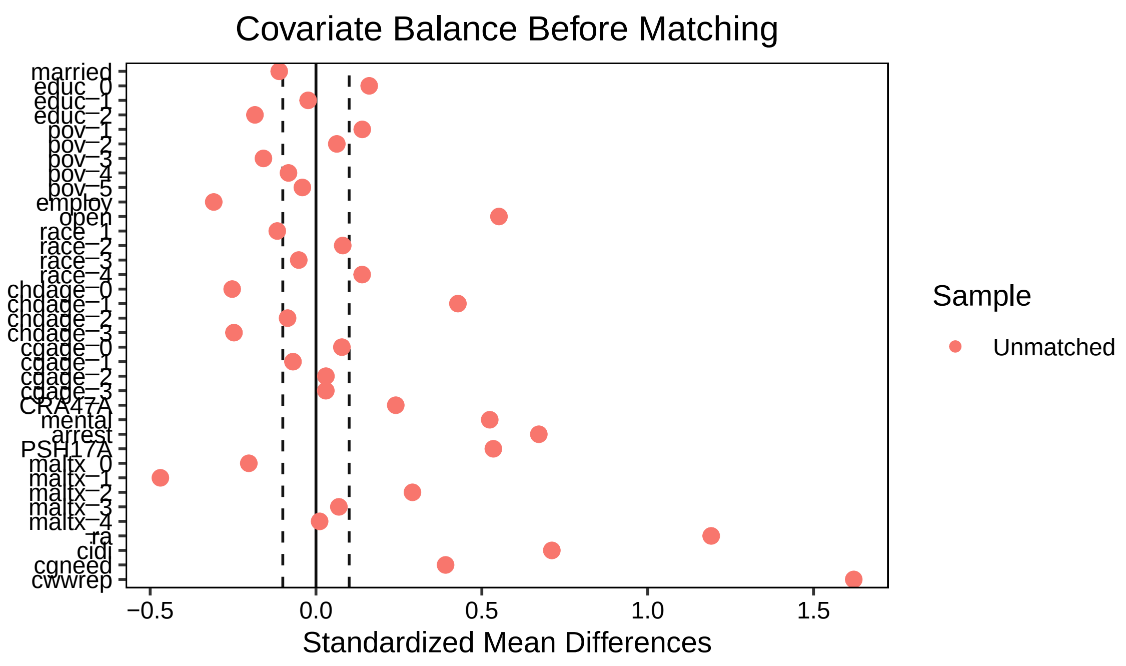
```r
# Balance table
cobalt::bal.tab(
  select(
    gm_df, married, educ, pov, employ, open, race, chdage, cgage, CRA47A,
    mental, arrest, PSH17A, maltx, ra, cidi, cgneed, cwwrep
  ),
  treat = gm_df$aodserv,
  s.d.denom = "treated", # ATT
  threshold = .1,
  stats = "mean.diff",
  continuous = "std",
  binary = "std"
)
```

```
## Balance Measures
##             Type Diff.Un     M.Threshold.Un
## married   Binary -0.1108 Not Balanced, >0.1
## educ_0    Binary  0.1605 Not Balanced, >0.1
## educ_1    Binary -0.0236     Balanced, <0.1
## educ_2    Binary -0.1840 Not Balanced, >0.1
## pov_1     Binary  0.1395 Not Balanced, >0.1
## pov_2     Binary  0.0628     Balanced, <0.1
## pov_3     Binary -0.1584 Not Balanced, >0.1
## pov_4     Binary -0.0829     Balanced, <0.1
## pov_5     Binary -0.0410     Balanced, <0.1
## employ    Binary -0.3082 Not Balanced, >0.1
## open      Binary  0.5516 Not Balanced, >0.1
## race_1    Binary -0.1167 Not Balanced, >0.1
## race_2    Binary  0.0807     Balanced, <0.1
## race_3    Binary -0.0519     Balanced, <0.1
## race_4    Binary  0.1392 Not Balanced, >0.1
## chdage_0 Binary -0.2527 Not Balanced, >0.1
## chdage_1 Binary  0.4279 Not Balanced, >0.1
## chdage_2 Binary -0.0856     Balanced, <0.1
## chdage_3 Binary -0.2473 Not Balanced, >0.1
## cgage_0   Binary  0.0781     Balanced, <0.1
## cgage_1   Binary -0.0694     Balanced, <0.1
## cgage_2   Binary  0.0300     Balanced, <0.1
## cgage_3   Binary  0.0297     Balanced, <0.1
## CRA47A    Binary  0.2406 Not Balanced, >0.1
## mental    Binary  0.5238 Not Balanced, >0.1
## arrest    Binary  0.6718 Not Balanced, >0.1
## PSH17A    Binary  0.5347 Not Balanced, >0.1
## maltx_0   Binary -0.2025 Not Balanced, >0.1
## maltx_1   Binary -0.4690 Not Balanced, >0.1
## maltx_2   Binary  0.2909 Not Balanced, >0.1
## maltx_3   Binary  0.0690     Balanced, <0.1
```

[11]Stuart, E. A., Lee, B. K., & Leacy, F. P. (2013). Prognostic score–based balance measures for propensity score methods in comparative effectiveness research. *Journal of Clinical Epidemiology*, 66(8 0), S84-S90.e1. https://doi.org/10.1016/j.jclinepi.2013.01.013

```
## maltx_4  Binary  0.0110     Balanced, <0.1
## ra       Binary  1.1915 Not Balanced, >0.1
## cidi     Binary  0.7110 Not Balanced, >0.1
## cgneed   Binary  0.3907 Not Balanced, >0.1
## cwwrep   Binary  1.6213 Not Balanced, >0.1
##
## Balance tally for mean differences
##                      count
## Balanced, <0.1          13
## Not Balanced, >0.1      23
##
## Variable with the greatest mean difference
##  Variable Diff.Un     M.Threshold.Un
##    cwwrep  1.6213 Not Balanced, >0.1
##
## Sample sizes
##     Control Treated
## All    2460     298
```

```
# Love plot
cobalt::love.plot(
  select(
    gm_df, married, educ, pov, employ, open, race, chdage, cgage, CRA47A,
    mental, arrest, PSH17A, maltx, ra, cidi, cgneed, cwwrep
  ),
  treat = gm_df$aodserv,
  binary = "std",
  s.d.denom = "treated",
  threshold = .1,
  sample.names = c("Unmatched")
) +
  labs(title = "Covariate Balance Before Matching")
```



Greedy Nearest Neighbor Matching Without Replacement

By default, the `MatchIt::matchit()` function performs greedy nearest neighbor matching without replacement, therefore the `method = "nearest"` and `replace = F` arguments do not need to be specified. Non-replacement means that once a treated case is matched to a non-treated case, both cases are removed from the pool. Matching with replacement allows each control unit to be matched with any number of treated units.

To avoid dissimilar matches, we can constrain matches so that the absolute distance of propensity scores between two participants is less than a specified tolerance for matching or a caliper. The width of the caliper is by default in standard deviation units and can be specified using the `caliper` argument. A wide caliper may result in more matches and a larger sample, but inexact matching may occur as indicated by large distances on the propensity score between the treated and nontreated cases. Using varying caliper sizes can test the sensitivity of the findings. Here we use a caliper size of a quarter of a standard deviation, which is suggested by Rosenbaum and Rubin (1985). Austin (2011) recommends a caliper size of 0.2 of the standard deviation.[12]

The order of the matching can be specified using the `m.order` argument. If this argument is set to `largest`, then matching begins with the treated subject with the highest propensity score; if set to `smallest`, then matching takes places in ascending order of the distance measures; and if `random`, matching takes place in a random order. "The default is to go in descending order from the highest propensity score; doing so allows the units that would have the hardest time finding close matches to be matched first."[13] Note that when non-treated cases have the same propensity score values, their matches will depend on the order of the data. Thus, it is still important to randomly shuffle your data prior to matching.

Finally, the logit of the predicted probability from a logistic regression model can be supplied to the `distance` argument. The logit of the predicted probability is used, because the logit is approximately normally distributed. Matching on the logit also improves balance, compared to matching on the raw propensity score. To calculate this automatically, set `distance="glm"` and `link="logit"`.

The choice of explanatory variables (i.e., conditioning variables) in the model predicting propensity scores of service receipt serves a paramount role in the propensity score analysis. We chose these variables based on a review of substance abuse literature to determine what characteristics were associated with treatment receipt:

```r
# Logistic regression specification
(gm_f <- cobalt::f.build("aodserv", select(gm_df, PSH17A:other, -aodserv)))
```

```
## aodserv ~ PSH17A + CRA47A + married + high + bahigh + poverty2 +
##     poverty3 + poverty4 + poverty5 + employ + open + black +
##     hispanic + natam + cgrage1 + cgrage2 + cgrage3 + chdage1 +
##     chdage2 + chdage3 + mental + arrest + sexual + provide +
##     supervis + other
## <environment: 0x000001cfc6c35578>
```

We then calculate the logit of the predicted probability as the propensity score:

```r
gm_psm <- glm(gm_f, data = gm_df, family = binomial)
gm_ps <- predict(gm_psm, newdata = gm_df, type = "response")
gm_ps_logit <- log((1 - gm_ps) / gm_ps)
```

And then perform greedy nearest neighbor matching without replacement

[12] https://pubmed.ncbi.nlm.nih.gov/20925139/
[13] https://kosukeimai.github.io/MatchIt/articles/matching-methods.html

```r
set.seed(1000)
(gm_out <- MatchIt::matchit(
  gm_f,
  data = gm_df,
  distance = gm_ps_logit,
  m.order = "largest", # descending order
  caliper = .25
))
```

```
## A ‘matchit‘ object
##  - method: 1:1 nearest neighbor matching without replacement
##  - distance: User-defined [caliper]
##  - caliper: <distance> (0.311)
##  - number of obs.: 2758 (original), 566 (matched)
##  - target estimand: ATT
##  - covariates: PSH17A, CRA47A, married, high, bahigh, poverty2, poverty3, poverty4, poverty5, employ
```

Notice that a limitation of using calipers is that only 283 out of 298 of the original treated cases were matched. This matching scheme reduces the sample size from 2758 to 566—283 cases in the control group and 283 cases in the treated group.

```r
set.seed(1000)
(gm_out_20 <- MatchIt::matchit(
  gm_f,
  data = gm_df,
  distance = gm_ps_logit,
  m.order = "largest", # descending order
  caliper = .20
))
```

```
## A ‘matchit‘ object
##  - method: 1:1 nearest neighbor matching without replacement
##  - distance: User-defined [caliper]
##  - caliper: <distance> (0.249)
##  - number of obs.: 2758 (original), 566 (matched)
##  - target estimand: ATT
##  - covariates: PSH17A, CRA47A, married, high, bahigh, poverty2, poverty3, poverty4, poverty5, employ
```

The `matchit` object will return a `match.matrix`, which contains the treated units as the rownames and the values in each row the names or indices of the control units matched to the treated units:

```r
head(gm_out$match.matrix)
```

```
##    [,1]
## 2  "420"
## 3  "1852"
## 10 "2130"
## 14 NA
## 41 "782"
## 49 "1162"
```

Remember to use weights when estimating the treatment effect (but this is not necessary when 1:1 matching without replacement was performed):

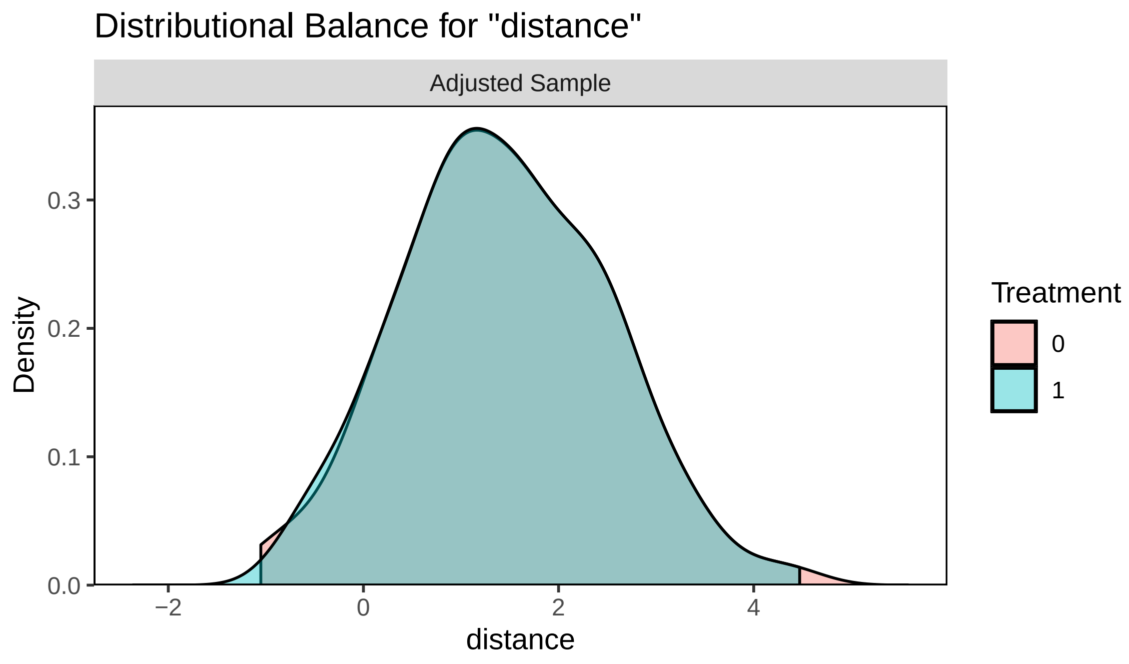
```r
head(gm_out$weights)
```

```
## 1 2 3 4 5 6
## 0 1 1 0 0 0
```

Check Common Support Greedy matching is criticized, because it requires a sizeable common-support region to work. The common support region is defined as the region bounded by the maximum value of estimated propensity scores for the treated participants and by the minimum value of the estimated propensity scores for the nontreated participants. In this example, a sizeable common-support region exists. The `discard` argument in `matchit()` can be used to discard units outside a region of common support.

```r
cobalt::bal.plot(gm_out, var.name = "distance")
```

```
## Ignoring unknown labels:
## * colour : "Treatment"
```



Check Balance Covariate balance can be assessed using hypothesis tests, such as `chisq.test`:

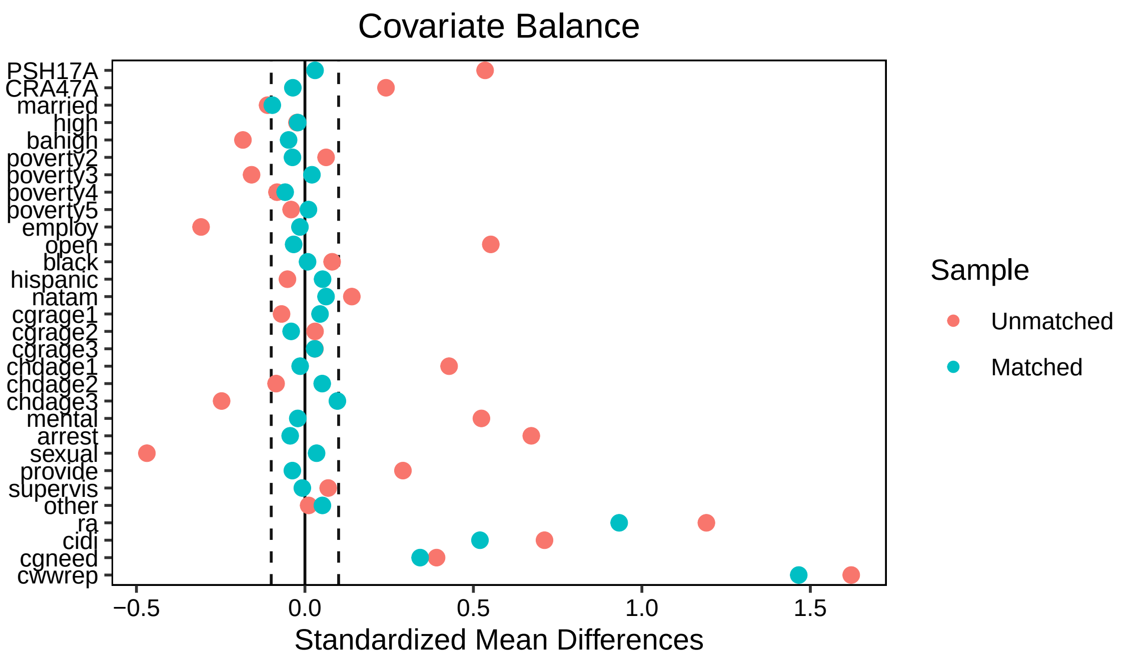
```r
# Extract matched data
gm_out_data <- MatchIt::match.data(gm_out)

# Assess balance on "ra"
chisq.test(gm_out_data$ra, gm_out_data$aodserv)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
```

```
##
## data:  gm_out_data$ra and gm_out_data$aodserv
## X-squared = 108.05, df = 1, p-value < 0.00000000000000022
```

The object from `matchit()` can be directly used in `cobalt` functions to produce balance tables and plots. To specify additional variables for which to display balance, use the argument `addl` in conjunction with the `data` argument.

```
cobalt::love.plot(gm_out,
  binary = "std", threshold = c(m = .1), drop.distance = T,
  addl = c("ra", "cidi", "cgneed", "cwwrep"), data = gm_df,
  sample.names = c("Unmatched", "Matched")
)
```



```
cobalt::bal.tab(gm_out,
  binary = "std", threshold = c(m = .1), un = T,
  addl = c("ra", "cidi", "cgneed", "cwwrep"), data = gm_df
)
```

```
## Balance Measures
##               Type Diff.Un Diff.Adj       M.Threshold
## distance Distance -1.3021   0.0068     Balanced, <0.1
## PSH17A     Binary  0.5347   0.0301     Balanced, <0.1
## CRA47A     Binary  0.2406  -0.0359     Balanced, <0.1
## married    Binary -0.1108  -0.0969     Balanced, <0.1
## high       Binary -0.0236  -0.0214     Balanced, <0.1
## bahigh     Binary -0.1840  -0.0485     Balanced, <0.1
## poverty2   Binary  0.0628  -0.0370     Balanced, <0.1
## poverty3   Binary -0.1584   0.0207     Balanced, <0.1
## poverty4   Binary -0.0829  -0.0587     Balanced, <0.1
## poverty5   Binary -0.0410   0.0105     Balanced, <0.1
## employ     Binary -0.3082  -0.0148     Balanced, <0.1
## open       Binary  0.5516  -0.0335     Balanced, <0.1
## black      Binary  0.0807   0.0078     Balanced, <0.1
## hispanic   Binary -0.0519   0.0524     Balanced, <0.1
## natam      Binary  0.1392   0.0626     Balanced, <0.1
## cgrage1    Binary -0.0694   0.0448     Balanced, <0.1
## cgrage2    Binary  0.0300  -0.0409     Balanced, <0.1
## cgrage3    Binary  0.0297   0.0289     Balanced, <0.1
## chdage1    Binary  0.4279  -0.0142     Balanced, <0.1
## chdage2    Binary -0.0856   0.0513     Balanced, <0.1
## chdage3    Binary -0.2473   0.0963     Balanced, <0.1
## mental     Binary  0.5238  -0.0212     Balanced, <0.1
## arrest     Binary  0.6718  -0.0438     Balanced, <0.1
## sexual     Binary -0.4690   0.0346     Balanced, <0.1
## provide    Binary  0.2909  -0.0373     Balanced, <0.1
## supervis   Binary  0.0690  -0.0077     Balanced, <0.1
## other      Binary  0.0110   0.0518     Balanced, <0.1
## ra         Binary  1.1915   0.9327 Not Balanced, >0.1
## cidi       Binary  0.7110   0.5195 Not Balanced, >0.1
## cgneed     Binary  0.3907   0.3419 Not Balanced, >0.1
## cwwrep     Binary  1.6213   1.4656 Not Balanced, >0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1        27
## Not Balanced, >0.1     4
##
## Variable with the greatest mean difference
##  Variable Diff.Adj        M.Threshold
##    cwwrep   1.4656 Not Balanced, >0.1
##
## Sample sizes
##           Control Treated
## All          2460     298
## Matched       283     283
## Unmatched    2177      15
```

Greedy Nearest Neighbor Mahalanobis Distance Matching Without Replacement

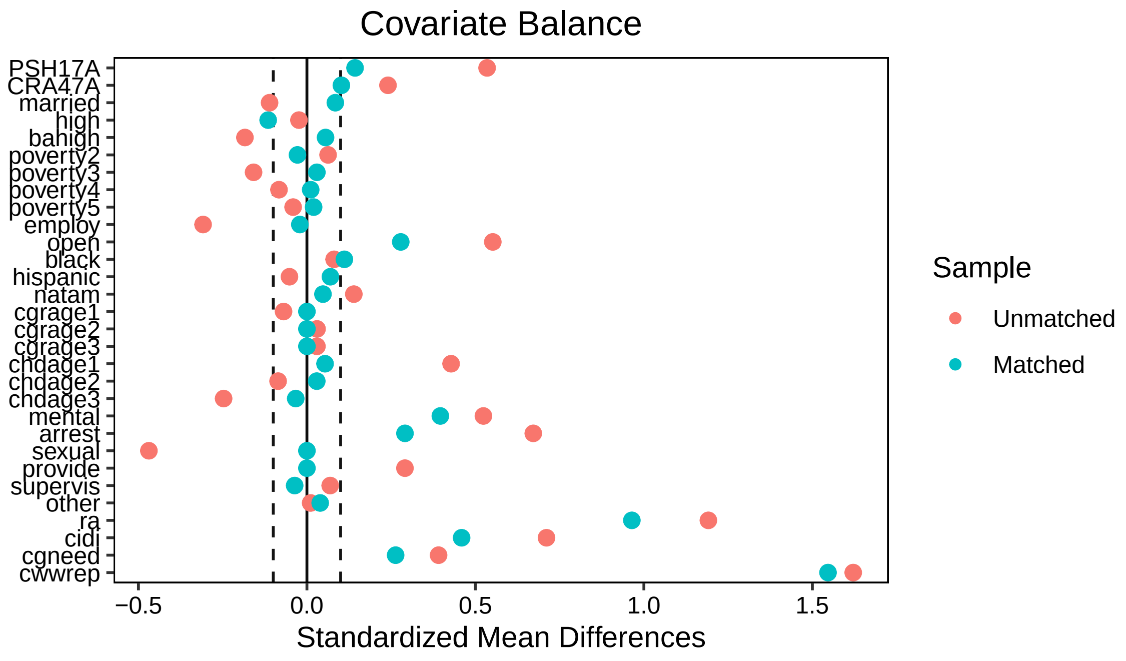
Here we perform Mahalanobis distance matching without replacement and without including estimated propensity scores by setting the `distance` argument to "mahalanobis." If propensity scores need to be estimated, the argument `mahvars` can be used to specify the variables used to create the Mahalanobis distance, while the `distance` argument should be either a vector of the propensity scores or whatever method is desired for estimating the propensity scores.

```
set.seed(1000)
(gm_out2 <- MatchIt::matchit(
  gm_f,
  data = gm_df,
  method = "nearest",
  distance = "mahalanobis",
  replace = F
))
```

```
## A ‘matchit‘ object
##  - method: 1:1 nearest neighbor matching without replacement
##  - distance: Mahalanobis - number of obs.: 2758 (original), 596 (matched)
##  - target estimand: ATT
##  - covariates: PSH17A, CRA47A, married, high, bahigh, poverty2, poverty3, poverty4, poverty5, employ
```

```
cobalt::love.plot(gm_out2,
  binary = "std", threshold = c(m = .1),
  addl = c("ra", "cidi", "cgneed", "cwwrep"), data = gm_df,
  sample.names = c("Unmatched", "Matched")
)
```

Check Balance



```
cobalt::bal.tab(gm_out2,
  binary = "std", threshold = c(m = .1),
  addl = c("ra", "cidi", "cgneed", "cwwrep"), data = gm_df
)
```

```
## Balance Measures
##             Type Diff.Adj        M.Threshold
## PSH17A    Binary   0.1429 Not Balanced, >0.1
## CRA47A    Binary   0.1022 Not Balanced, >0.1
## married   Binary   0.0843     Balanced, <0.1
## high      Binary  -0.1150 Not Balanced, >0.1
## bahigh    Binary   0.0552     Balanced, <0.1
## poverty2 Binary  -0.0281     Balanced, <0.1
## poverty3 Binary   0.0295     Balanced, <0.1
## poverty4 Binary   0.0112     Balanced, <0.1
## poverty5 Binary   0.0199     Balanced, <0.1
## employ    Binary  -0.0211     Balanced, <0.1
## open      Binary   0.2784 Not Balanced, >0.1
## black     Binary   0.1111 Not Balanced, >0.1
## hispanic Binary   0.0696     Balanced, <0.1
## natam     Binary   0.0476     Balanced, <0.1
## cgrage1   Binary   0.0000     Balanced, <0.1
## cgrage2   Binary   0.0000     Balanced, <0.1
## cgrage3   Binary   0.0000     Balanced, <0.1
## chdage1   Binary   0.0538     Balanced, <0.1
## chdage2   Binary   0.0292     Balanced, <0.1
## chdage3   Binary  -0.0333     Balanced, <0.1
## mental    Binary   0.3960 Not Balanced, >0.1
## arrest    Binary   0.2910 Not Balanced, >0.1
## sexual    Binary   0.0000     Balanced, <0.1
## provide   Binary   0.0000     Balanced, <0.1
## supervis Binary  -0.0364     Balanced, <0.1
## other     Binary   0.0394     Balanced, <0.1
## ra        Binary   0.9643 Not Balanced, >0.1
## cidi      Binary   0.4591 Not Balanced, >0.1
## cgneed    Binary   0.2633 Not Balanced, >0.1
## cwwrep    Binary   1.5465 Not Balanced, >0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1        19
## Not Balanced, >0.1    11
##
## Variable with the greatest mean difference
##  Variable Diff.Adj        M.Threshold
##    cwwrep   1.5465 Not Balanced, >0.1
##
## Sample sizes
##           Control Treated
## All          2460     298
## Matched       298     298
## Unmatched    2162       0
```

As seen above, balance has not been achieved in multiple covariates. According to Stuart (2010), "the

Mahalanobis distance can work quite well when there are relatively few covariates (fewer than 8), but it does not perform as well when the covariates are not normally distributed or there are many covariates."[14]

Extract Matched Data After matching, the treatment effect can be estimated using the matched sample, which can be extracted using the `MatchIt::match.data()` function.

```r
MatchIt::match.data(gm_out2)
```

[14]Stuart, E. A. (2010). Matching Methods for Causal Inference: A Review and a Look Forward. *Statistical Science*, *25*(1), 1–21. https://doi.org/10.1214/09-STS313

Propensity Score Estimation Using Generalized Boosted Regression

Generalized boosted regression is an iterative method for creating propensity scores. It uses an automated, data-adaptive algorithm that fits several models by way of a regression tree, and then merges the predictions produced by each model. The regression tree partitions the sample into small groups based on predictor variables.

GBR is a sum of regression trees. These trees are computationally fast to fit, and they are invariant to one-to-one transformations of the independent variables. Its advantage over logistic regression is that it doesn't require knowing the functional form of predictor variables.

Load Package

Generalized boosted regression (GBR) requires the `gbm` package.

```
pacman::p_load(gbm)
```

Load Data and Sort

After importing the data, missing data is deleted listwise, and the data is sorted randomly. According to the `gbm` package vignette, if the data is sorted in a systematic way, then the data should be shuffled before running `gbm`. To create reproducible results, we need to use the `set.seed()` function.

```
set.seed(1000)
gbr_df <- read_dta(here("data", "g3aca1.dta")) %>%
  haven::zap_formats() %>%
  sjlabelled::remove_all_labels() %>%
  as_tibble() %>%
  select(
    intbl, ageyc, fmale, blck, whit, hisp, pcedu, ipovl, pcemft, fthr,
    dicsagg2, dicsint2, dccereg2, dccscom2, dccpros2, draggr2
  ) %>%
  drop_na() %>%
  add_column(runif = runif(nrow(.))) %>%
  arrange(runif) %>%
  select(-runif)
```

Fit Generalized Boosted Regression Model

The `gbm::gbm()` function has many arguments that can be fine-tuned. See `?gbm` for a detailed description of each argument. For example, to reduce prediction error, the `train.fraction` argument can be set to use a subsample of the observations for the estimation process. Here we also use `interaction.depth = 4` to specify a maximum of four splits for each sample tree used in the model, which allows all four-way interactions between all covariates to be considered for optimizing the likelihood function at each iteration. The `shrinkage` argument is also known as the learning rate or step-size reduction; we use a value of .0005 to ensure a smooth fit.

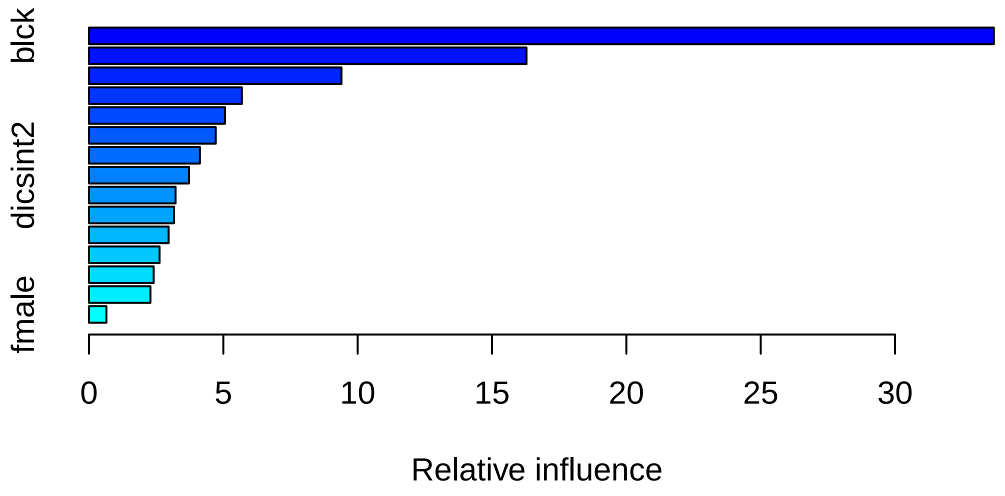
A summary of the fitted model provides us with *relative influence*, which is the percentage of log likelihood explained by each input variable. The percentages of influence for all predictor variables sum to 100%.

The GBM ouput showed that `blck` had the strongest influence on the likelihood function (33.7%), followed by `ageyc` (16.3%) and `draggr2` (9.4%).

```r
(gbr_f <- cobalt::f.build("intbl", select(gbr_df, -intbl)))
```

```
## intbl ~ ageyc + fmale + blck + whit + hisp + pcedu + ipovl +
##     pcemft + fthr + dicsagg2 + dicsint2 + dccereg2 + dccscom2 +
##     dccpros2 + draggr2
## <environment: 0x000001cfbeba1b60>
```

```r
set.seed(1000)
gbr_m1 <- gbm::gbm(
  formula = gbr_f,
  data = gbr_df,
  distribution = "bernoulli",
  n.trees = 1000, # number of trees to fit
  train.fraction = 0.8, # a random 80% subsample for estimation
  interaction.depth = 4, # allow all four-way interactions
  shrinkage = 0.0005 # small shrinkage to ensure smooth fit
)
summary(gbr_m1)
```



```
##                var    rel.inf
## blck          blck 33.6768868
## ageyc        ageyc 16.2834103
## draggr2    draggr2  9.3962151
## whit          whit  5.6895119
## ipovl        ipovl  5.0614691
## pcemft      pcemft  4.7179839
## pcedu        pcedu  4.1280063
## dicsint2 dicsint2  3.7223185
## dicsagg2 dicsagg2  3.2228567
## dccscom2 dccscom2  3.1648839
## dccereg2 dccereg2  2.9647178
## dccpros2 dccpros2  2.6269129
## hisp          hisp  2.4079157
## fthr          fthr  2.2874508
## fmale        fmale  0.6494603
```

Estimate Propensity Scores

After fitting the model, estimate propensity scores using the `predict.gbm()` function.

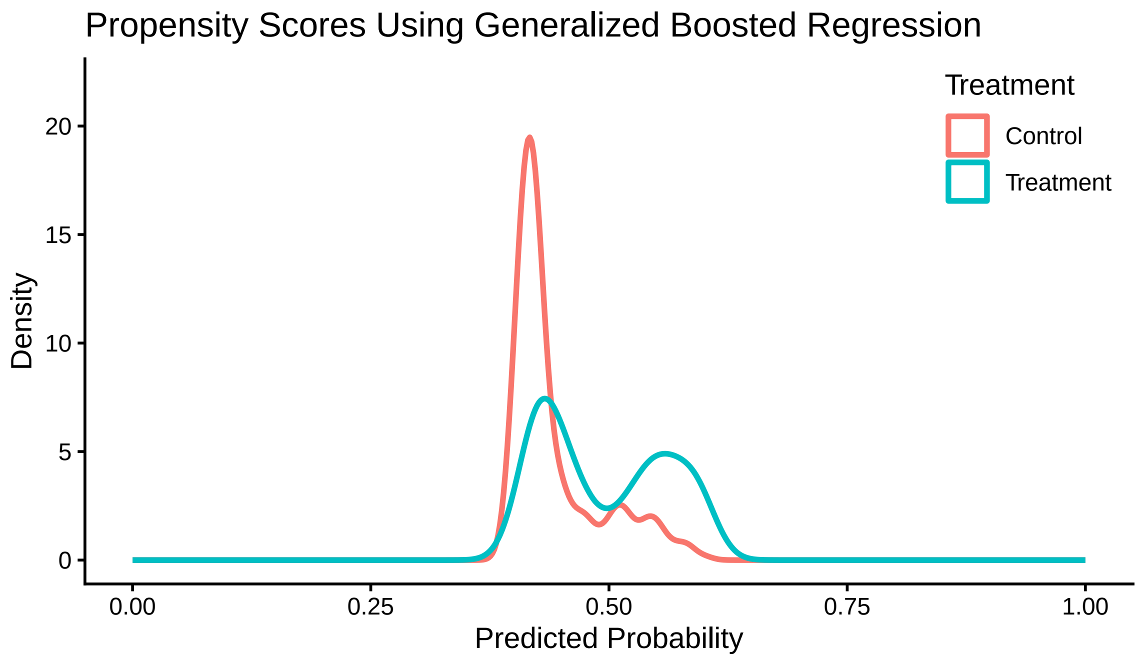
```
psb <- gbm::predict.gbm(gbr_m1, data = gbr_df, type = "response")
head(psb)
```

```
## [1] 0.4441531 0.4405950 0.5288983 0.5958331 0.3985049 0.5562076
```

Plot Propensity Score Distributions

As seen in the figure below, the propensity scores estimated by GBM has some overlap between the control and treatment groups (i.e., “common support”).

```r
gbr_df %>%
  mutate(psb = psb, intbl = factor(intbl, labels = c("Control", "Treatment"))) %>%
  ggplot(aes(x = psb, color = intbl)) +
  theme_classic() +
  geom_density(linewidth = 1) +
  xlim(0, 1) +
  ylim(0, 22) +
  labs(
    x = "Predicted Probability", y = "Density",
    title = "Propensity Scores Using Generalized Boosted Regression",
    color = "Treatment"
  ) +
  theme(legend.position = c(0.9, 0.85))
```



Summary Statistics of Propensity Scores

```r
summary(psb)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.3913  0.4189  0.4383  0.4674  0.5202  0.6082
```

GBR Using the WeightIt Package

As an alternative to the `gbm` package, the `WeightIt` package can also fit GBR models. Note that the default maximum number of trees used (the `n.trees` argument) is 10000 for binary treatments.

```r
set.seed(1000)
(gbr_m2 <- WeightIt::weightit(
  formula = gbr_f,
  data = gbr_df,
  method = "gbm",
  estimand = "ATE",
  distribution = "bernoulli", # binary treatment
  stop.method = "es.mean",
  n.trees = 10000,
  nTrain = 0.8 * nrow(gbr_df),
  interaction.depth = 4,
  shrinkage = 0.0005
))
```

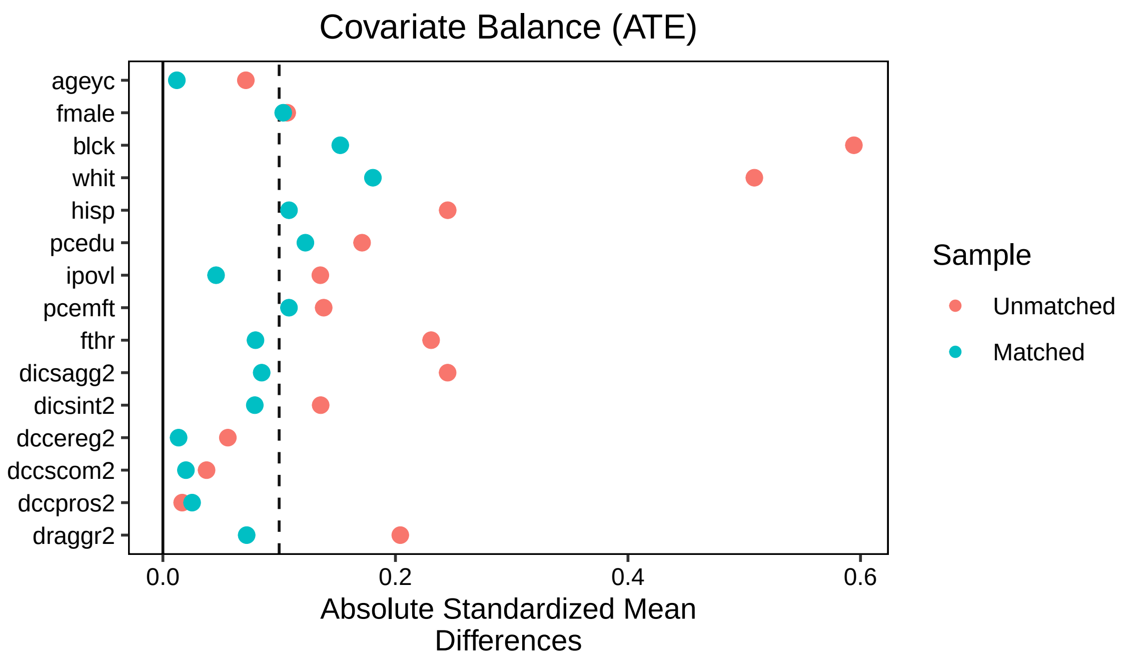
```
## A weightit object
##  - method: "gbm" (propensity score weighting with GBM)
##  - number of obs.: 603
##  - sampling weights: none
##  - treatment: 2-category
##  - estimand: ATE
##  - covariates: ageyc, fmale, blck, whit, hisp, pcedu, ipovl, pcemft, fthr, dicsagg2, dicsint2, dccer
```

```r
# Propensity Scores
head(gbr_m2$ps)
```

```
## [1] 0.3781517 0.6075811 0.6596871 0.7766516 0.2138153 0.7927817
```

Check Balance Using a standardized mean difference cut-off point of 0.1, it can be seen below that balance has been achieved in most, but not all, of the covariates:

```r
cobalt::love.plot(gbr_m2,
  thresholds = c(m = .1),
  binary = "std", abs = T, drop.distance = T,
  sample.names = c("Unmatched", "Matched")
) +
  labs(title = "Covariate Balance (ATE)")
```



Other Packages

GBR is also implemented in the `MatchIt::matchit()` and `twang::ps()` functions. While the `gbm` package is used in both, each function uses a different set of default arguments. See their respective help files for more details.

4.7 Computer Lab: Imbalance Check and Outcome Analysis with R after optmatch

Optimal Matching

Load Packages

```
pacman::p_load(optmatch, knitr, broom, sandwich, rlang, marginaleffects)
select <- dplyr::select
```

Load Data

This dataset comes from a study that investigates intergenerational dependence on welfare and its relation to child academic achievement.

The dependent variable is `lwss97`, the age-normed "letter-word" identification score of the Woodcock-Johnson Revised Tests of Achievement. A high score on this measure indicates high achievement. The treatment variable is `kuse` or children who ever used Aid to Families With Dependent Children (AFDC). The covariates are:

- `mratio96`: Ratio of Family Income to Poverty Line in 1996
- `pcged97`: Caregiver's Education in 1997 (Years of Schooling)
- `pcg_adc`: Caregiver's History of Using Welfare (Number of Years; range: 0-7)
- `black`: Child's Race: African American (1 = African American; 0 = Other)
- `age97`: Child's Age in 1997
- `male`: Child's Gender: Male (1 = Male; 0 = Female)

```
d <- haven::read_dta("data/opt/chpt5_2_original.dta")
df <- haven::read_dta("data/opt/chpt5_2.dta")
cds <- haven::read_dta("data/opt/chpt5_2ps.dta")
```

Bivariate Comparisons

With the exception of child's gender, the difference on each covariate between treated and control groups is statistically significant. Without controlling for these covariates, the study's estimate of treatment effect would be biased.

```
# Intergenerational dependence
chisq.test(d$kuse, d$puse, correct = F)
```

```
##
##  Pearson's Chi-squared test
##
## data:  d$kuse and d$puse
## X-squared = 88.274, df = 1, p-value < 0.00000000000000022
```

```r
# Wilcoxon Rank-Sum (Mann-Whitney) test and t-test
d %>%
  select(mratio96, pcged97, pcg_adc, black, age97, male, kuse) %>%
  pivot_longer(-kuse, names_to = "variable") %>%
  group_by(variable) %>%
  nest() %>%
  mutate(wilcoxon = map(data, ~ wilcox.test(.$value ~ .$kuse, correct = F))) %>%
  mutate(wilcoxon.stat = map(wilcoxon, ~ round(.$statistic, 3))) %>%
  mutate(wilcoxon.pvalue = map(wilcoxon, ~ round(.$p.value, 3))) %>%
  unnest(cols = c(wilcoxon.stat, wilcoxon.pvalue)) %>%
  mutate(ttest = map(data, ~ t.test(.$value ~ .$kuse, var.equal = T))) %>%
  mutate(t.stat = map(ttest, ~ round(.$statistic, 3))) %>%
  mutate(t.pvalue = map(ttest, ~ round(.$p.value, 3))) %>%
  unnest(cols = c(t.stat, t.pvalue)) %>%
  select(-data, -wilcoxon, -ttest)
```

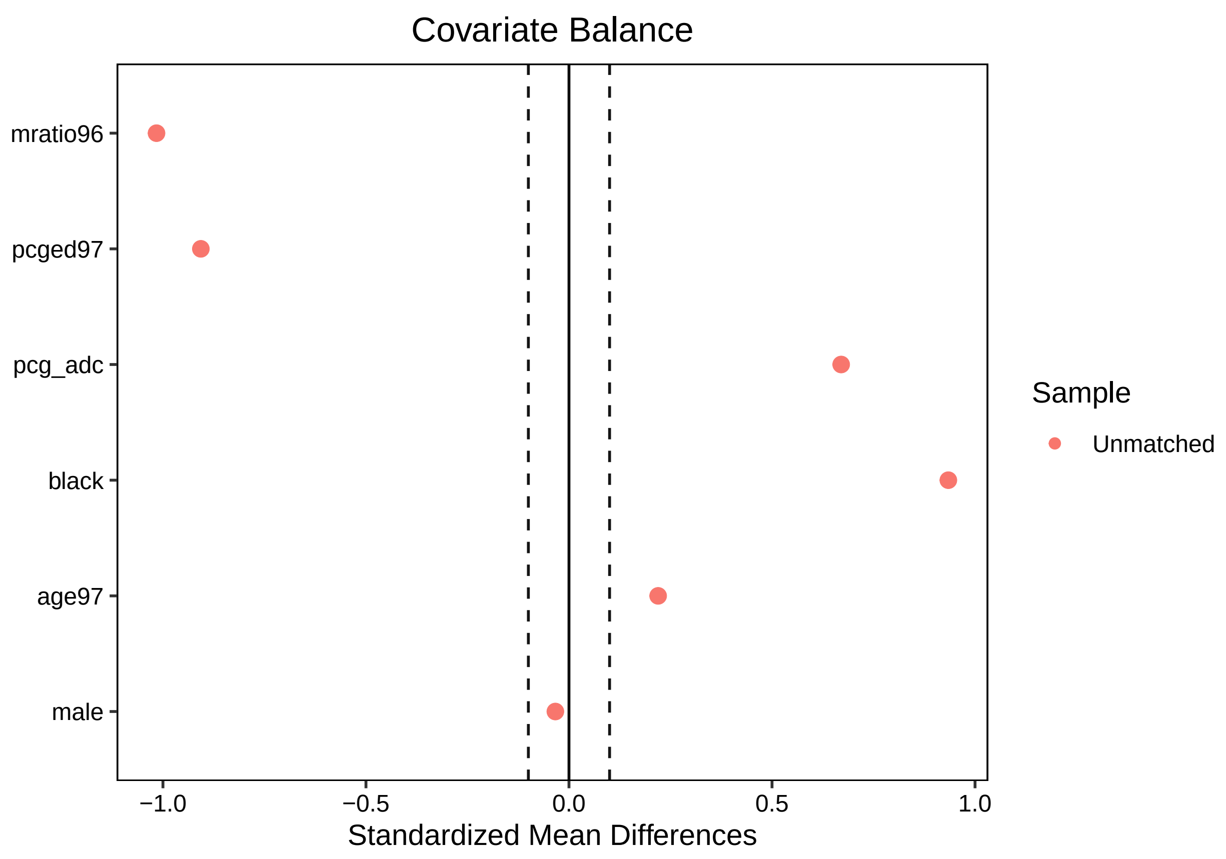
```
## # A tibble: 6 x 5
## # Groups:   variable [6]
##   variable wilcoxon.stat wilcoxon.pvalue  t.stat t.pvalue
##   <chr>            <dbl>           <dbl>   <dbl>    <dbl>
## 1 mratio96       172618.           0      12.3      0
## 2 pcged97        147410.           0      12.3      0
## 3 pcg_adc         69728            0     -11.0      0
## 4 black           58038            0     -12.7      0
## 5 age97           87129            0.002  -3.11     0.002
## 6 male           101544            0.636   0.473    0.637
```

```r
# SMD Balance Checks
(c1 <- cobalt::bal.tab(
  x = d %>% select(mratio96, pcged97, pcg_adc, black, age97, male),
  treat = d$kuse,
  binary = "std",
  threshold = c(m = .1),
  s.d.denom = "pooled"
))
```

```
## Balance Measures
##             Type Diff.Un     M.Threshold.Un
## mratio96 Contin. -1.0158 Not Balanced, >0.1
## pcged97  Contin. -0.9067 Not Balanced, >0.1
## pcg_adc  Contin.  0.6703 Not Balanced, >0.1
## black     Binary  0.9343 Not Balanced, >0.1
## age97    Contin.  0.2196 Not Balanced, >0.1
## male      Binary -0.0335     Balanced, <0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         1
## Not Balanced, >0.1     5
##
## Variable with the greatest mean difference
##  Variable Diff.Un     M.Threshold.Un
##  mratio96 -1.0158 Not Balanced, >0.1
```

```
##
## Sample sizes
##      Control Treated
## All      729     274
```

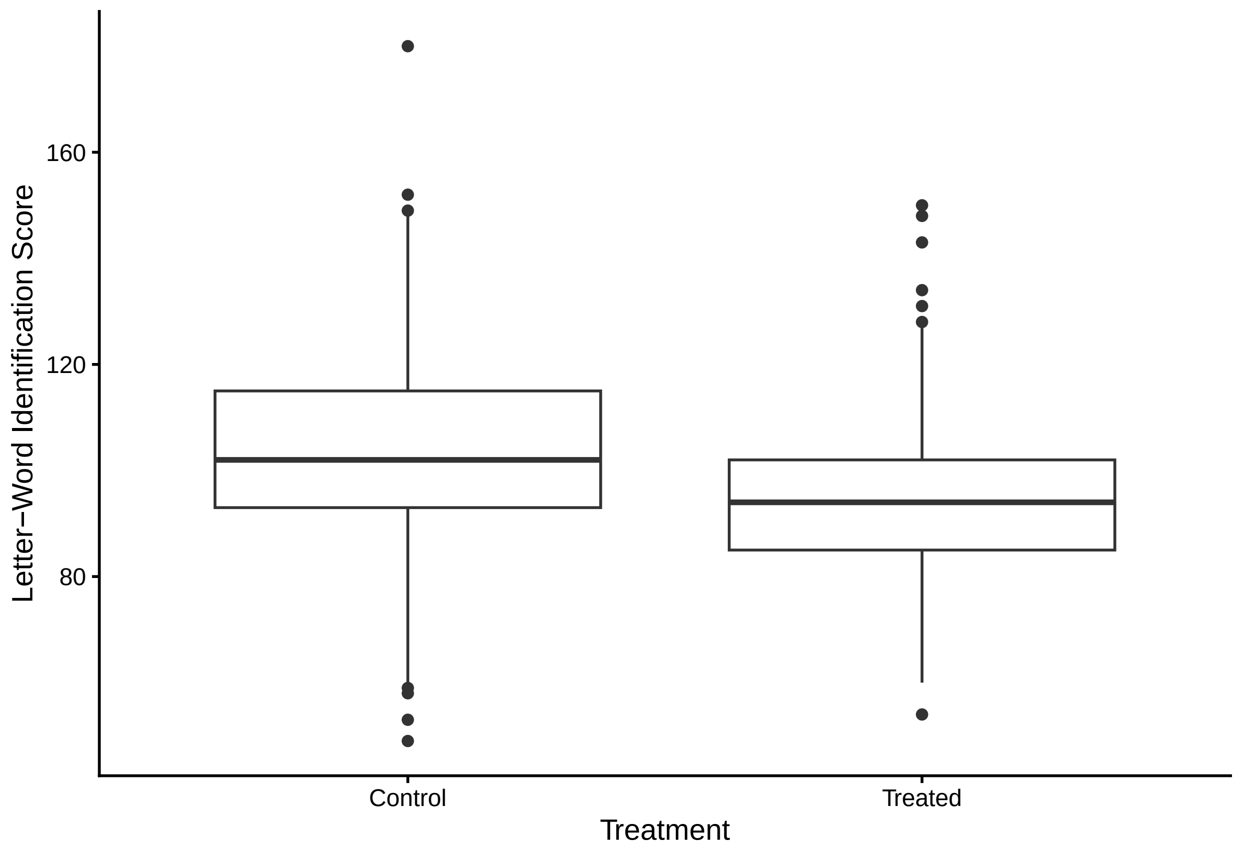
```
(cobalt::love.plot(c1, sample.names = c("Unmatched")))
```



Outcome Models Without Bias Control

We first observe that the control group had a higher letter-word identification score than the treatment group:

```
d %>%
  ggplot(aes(y = lwss97, x = factor(kuse))) +
  geom_boxplot() +
  labs(x = "Treatment", y = "Letter-Word Identification Score") +
  scale_x_discrete(labels = c("Control", "Treated")) +
  theme_classic()
```



We can estimate the ATE using an independent samples t test. It shows that the treated group on average had a mean letter-word identification score that was 9.82 points lower than that of the control group ($p < .001$).

```
t.test(lwss97 ~ kuse, data = d, var.equal = T) %>%
  broom::tidy()
```

```
## # A tibble: 1 x 10
##   estimate estimate1 estimate2 statistic  p.value parameter conf.low conf.high
##      <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl>
## 1     9.82      104.      94.2      8.51 6.27e-17      1001     7.56      12.1
## # i 2 more variables: method <chr>, alternative <chr>
```

We can also estimate the ATE using an OLS regression model. It shows that, controlling for covariates, the treated group on average had a letter-word identification score that is 4.73 points lower than that of the control group (p < .001).

```
reg <- lm(lwss97 ~ kuse + male + black + age97 + pcged97 + mratio96 + pcg_adc,
  data = d
)
broom::tidy(lmtest::coeftest(reg, vcov = sandwich::vcovCL, cluster = d$pcg_id))
```

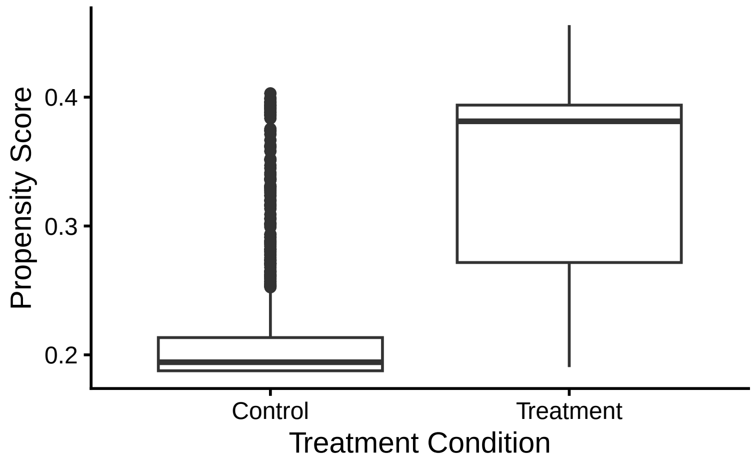
```
## # A tibble: 8 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   84.9       4.50      18.9  4.30e-68
## 2 kuse          -4.73      1.39      -3.40 7.00e- 4
## 3 male          -2.00      0.987     -2.02 4.34e- 2
## 4 black         -1.88      1.23      -1.53 1.27e- 1
## 5 age97          0.873     0.170      5.14 3.30e- 7
## 6 pcged97        0.910     0.336      2.71 6.91e- 3
## 7 mratio96       1.13      0.332      3.41 6.67e- 4
## 8 pcg_adc       -0.758     0.311     -2.44 1.50e- 2
```

Matching

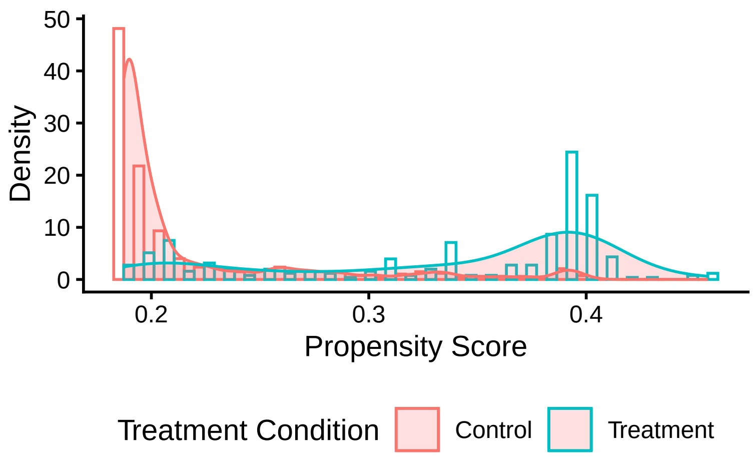
Note: To be consistent with the STATA result, we will use the propensity scores created by Stata.

Distribution of estimated propensity scores As seen in the figures below, there is no sizeable common support region of the estimated propensity scores by treatment status, so using greedy matching would produce a nontrivial loss of matched participants. Therefore, we decide to use optimal matching.

```
cds %>%
  ggplot(aes(x = factor(kuse), y = ps)) +
  geom_boxplot() +
  labs(y = "Propensity Score", x = "Treatment Condition") +
  scale_x_discrete(labels = c("Control", "Treatment")) +
  theme_classic()
```



```
cds %>%
  ggplot(aes(x = ps, color = factor(kuse))) +
  geom_histogram(aes(y = ..density..), fill = "white", position = "dodge") +
  geom_density(alpha = .2, fill = "#FF6666") +
  labs(y = "Density", x = "Propensity Score", color = "Treatment Condition") +
  scale_color_discrete(labels = c("Control", "Treatment")) +
  theme_classic() +
  theme(legend.position = "bottom")
```



We would lose 64.3% cases if we used nearest neighbor matching without replacement within specified caliper widths:

```
(nn_test <- matchit(kuse ~ mratio96 + pcged97 + pcg_adc + black + age97 + male,
  data = d,
  method = "nearest",
  distance = "glm",
  replace = F,
  caliper = .25
))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## A `matchit` object
##  - method: 1:1 nearest neighbor matching without replacement
##  - distance: Propensity score [caliper]
##
##              - estimated with logistic regression
##  - caliper: <distance> (0.07)
##  - number of obs.: 1003 (original), 358 (matched)
##  - target estimand: ATT
##  - covariates: mratio96, pcged97, pcg_adc, black, age97, male
```

```
(nrow(d) - nrow(match.data(nn_test))) / nrow(d)
```

```
## [1] 0.6430708
```

Optimal Matching The first argument that `optmatch::fullmatch()` requires is "a matrix of non-negative discrepancies, each indicating the permissibility and desirability of matching the unit corresponding to its row (treated) to the unit corresponding to its column (control)." First, we calculate the sample ranks of the propensity scores generated by generalized boosted regression using `rank(ps)`. Second, we use `outer(x, y, "-")`, which will subtract y control units from x treated units, to form a matrix with rows equal to the number of treated subjects and columns equal to the number of control subjects. Finally, we take the absolute value of these differences.

By default, `min.controls` or the minimum ratio of controls to treatments that is to be permitted within a matched set is zero, and `max.controls` or the maximum ratio of controls to treatments is infinite.

Hansen found that in the context of a specific application, variable matching with a specific structure worked best; that is, each treated participant was matched to at least $\frac{.5(1-\hat{P})}{\hat{P}}$ controls and at most $\frac{2(1-\hat{P})}{\hat{P}}$ controls, where $\hat{P}$ represents the proportion of treated participants in the sample. In this example, $\hat{P} = \frac{274}{729+274} = 27.32\%$, so the number of minimum controls is $\frac{.5(1-.2732)}{.2732} = 1.33$, and the number of maximum controls is $\frac{2(1-.2732)}{.2732} = 5.32$. There is a variance trade-off when selecting a variable number of controls, because the additional controls are generally less similar than the first closest match as the treated individual, which would increase bias, while the increase in sample size can decrease variance.[15]

For `optmatch::pairmatch()`, we can set the number of controls to be matched to each treatment using the `controls` argument.

```r
attach(cds)
prank <- rank(ps)
names(prank) <- kid
d1 <- outer(prank[kuse == 1], prank[kuse == 0], "-")
d1 <- abs(d1)
table(cds$kuse) # 729 control, 274 treated
dim(d1) # 274 x 729

# Full matching
fm <- fullmatch(d1)
(fm.d <- matched.distances(fm, d1, pres = TRUE))
max(unlist(fm.d))

# Variable Matching (at least 1, at most 4)
vm1 <- fullmatch(d1, min.controls = 1, max.controls = 4)
(vm1.d <- matched.distances(vm1, d1, pres = TRUE))
max(unlist(vm1.d))

# Variable Matching (at least 2, at most 4)
vm2 <- fullmatch(d1, min.controls = 2, max.controls = 4)
(vm2.d <- matched.distances(vm2, d1, pres = TRUE))
max(unlist(vm2.d))

# Variable Matching (using Hensen's equation)
vm3 <- fullmatch(d1, min.controls = 1.33, max.controls = 5.32)
(vm3.d <- matched.distances(vm3, d1, pres = TRUE))
max(unlist(vm3.d))

# Variable Matching (at least 2, at most 7)
vm4 <- fullmatch(d1, min.controls = 2, max.controls = 7)
```

[15]Stuart, E. A. (2010). Matching Methods for Causal Inference: A Review and a Look Forward. *Statistical Science*, *25*(1), 1–21. https://doi.org/10.1214/09-STS313

```
(vm4.d <- matched.distances(vm4, d1, pres = TRUE))
max(unlist(vm4.d))

# 1:1 Pair Matching
pm <- pairmatch(d1, controls = 1)
(pm.d <- matched.distances(pm, d1, pres = TRUE))
max(unlist(pm.d))
```

Optimal matching aims to minimize the total sample distance of propensity scores. We can extract the distances of matched units from their matched counterpairs using `optmatch::matched.distances()` and then calculate the total distance using `sum(unlist())`. The ratio of treatment to control and the number of strata for each structure can be inspected with `optmatch::stratumStructure()`.

```
mean(unlist(fm.d))
```

```
## [1] 35.89389
```

```
sum(unlist(fm.d))
```

```
## [1] 31120
```

```
stratumStructure(fm)
```

```
##  25:1  13:1  11:1  10:1   9:1   7:1   6:1   5:1   4:1   3:1   2:1   1:1   1:2
##     1     1     3     1     1     1     3     2     2     6     8    47    12
##   1:3   1:4   1:5   1:6   1:7   1:8   1:9  1:10  1:11  1:15  1:17  1:18  1:21
##     7     6     7     5     2     4     3     2     3     1     1     2     1
##  1:24  1:31  1:62 1:187
##     1     1     1     1
```

```
mean(unlist(vm1.d))
```

```
## [1] 328.7579
```

```
sum(unlist(vm1.d))
```

```
## [1] 239664.5
```

```
stratumStructure(vm1)
```

```
## 1:1 1:3 1:4
## 122   1 151
```

```
mean(unlist(vm2.d))
```

```
## [1] 348.1852
```

```r
sum(unlist(vm2.d))
```

```
## [1] 253827
```

```r
stratumStructure(vm2)
```

```
## 1:2 1:3 1:4
## 183   1  90
```

```r
mean(unlist(vm3.d))
```

```
## [1] 269.7222
```

```r
sum(unlist(vm3.d))
```

```
## [1] 196627.5
```

```r
stratumStructure(vm3)
```

```
## 1:1 1:6
## 183  91
```

```r
mean(unlist(vm4.d))
```

```
## [1] 312.3162
```

```r
sum(unlist(vm4.d))
```

```
## [1] 227678.5
```

```r
stratumStructure(vm4)
```

```
## 1:2 1:3 1:7
## 237   1  36
```

```r
mean(unlist(pm.d))
```

```
## [1] 143.2263
```

```r
sum(unlist(pm.d))
```

```
## [1] 39244
```

```r
stratumStructure(pm)
```

```
## 1:1 0:1
## 274 455
```

```r
# Define covariates
cov_labels <- tibble(
  cov = c(
    "mratio96", "pcged97", "pcg_adc",
    "black", "age97", "male"
  ),
  new = c(
    "Ratio of family income to poverty line in 1996",
    "Caregiver's education in 1997 (years of schooling)",
    "Caregiver's number of years using AFDC in childhood",
    "Child's race: African American (reference: other)",
    "Child's age in 1997", "Child's gender: male (reference: female)"
  ),
  order = c(1, 2, 3, 4, 5, 6)
)

# Calculate dx and dxm using imbalance()
arg1 <- rep(c("mratio96", "pcged97", "pcg_adc", "black", "age97", "male"), each = 7)
arg2 <- rep(c("before", "fm", "vm1", "vm2", "vm3", "vm4", "pm"), length(arg1) / 7)
table_5.10 <- map2_dfr(arg1, arg2, imbalance) %>%
  mutate(method = recode(method,
    `before` = "Before Matching",
    `fm` = "Full Matching",
    `vm1` = "Variable Matching 1 (at least 1, at most 4)",
    `vm2` = "Variable Matching 2 (at least 2, at most 4)",
    `vm3` = "Variable Matching 3 (Hansen's equation)",
    `vm4` = "Variable Matching 4 (at least 2, at most 7)",
    `pm` = "Pair matching"
  )) %>%
  left_join(cov_labels, by = "cov") %>%
  arrange(order) %>%
  select(cov, matching_scheme = method, dx, dxm) # or replace cov with new

# Print Table
print(table_5.10, n = 42)
```

Covariate Imbalance Before and After Matching by Matching Scheme

```
## # A tibble: 42 x 4
##    cov      matching_scheme                                  dx      dxm
##    <chr>    <chr>                                         <dbl>    <dbl>
##  1 mratio96 Before Matching                               1.02   NA
##  2 mratio96 Full Matching                                NA       0.0408
##  3 mratio96 Variable Matching 1 (at least 1, at most 4) NA       0.790
##  4 mratio96 Variable Matching 2 (at least 2, at most 4) NA       0.940
##  5 mratio96 Variable Matching 3 (Hansen’s equation)     NA       0.736
##  6 mratio96 Variable Matching 4 (at least 2, at most 7) NA       0.869
##  7 mratio96 Pair matching                                NA       0.253
##  8 pcged97  Before Matching                               0.907  NA
##  9 pcged97  Full Matching                                NA       0.168
## 10 pcged97  Variable Matching 1 (at least 1, at most 4) NA       0.722
## 11 pcged97  Variable Matching 2 (at least 2, at most 4) NA       0.867
## 12 pcged97  Variable Matching 3 (Hansen's equation)     NA       0.752
## 13 pcged97  Variable Matching 4 (at least 2, at most 7) NA       0.814
## 14 pcged97  Pair matching                               NA       0.340
## 15 pcg_adc  Before Matching                              0.670  NA
## 16 pcg_adc  Full Matching                               NA       0.00737
## 17 pcg_adc  Variable Matching 1 (at least 1, at most 4) NA       0.560
## 18 pcg_adc  Variable Matching 2 (at least 2, at most 4) NA       0.649
## 19 pcg_adc  Variable Matching 3 (Hansen's equation)     NA       0.551
## 20 pcg_adc  Variable Matching 4 (at least 2, at most 7) NA       0.626
## 21 pcg_adc  Pair matching                               NA       0.403
## 22 black    Before Matching                              0.933  NA
## 23 black    Full Matching                               NA       0.0106
## 24 black    Variable Matching 1 (at least 1, at most 4) NA       0.788
## 25 black    Variable Matching 2 (at least 2, at most 4) NA       0.919
## 26 black    Variable Matching 3 (Hansen's equation)     NA       0.732
## 27 black    Variable Matching 4 (at least 2, at most 7) NA       0.848
## 28 black    Pair matching                               NA       0.442
## 29 age97    Before Matching                              0.220  NA
## 30 age97    Full Matching                               NA       0.0690
## 31 age97    Variable Matching 1 (at least 1, at most 4) NA       0.215
## 32 age97    Variable Matching 2 (at least 2, at most 4) NA       0.219
## 33 age97    Variable Matching 3 (Hansen's equation)     NA       0.190
## 34 age97    Variable Matching 4 (at least 2, at most 7) NA       0.238
## 35 age97    Pair matching                               NA       0.137
## 36 male     Before Matching                              0.0335 NA
## 37 male     Full Matching                               NA       0.0532
## 38 male     Variable Matching 1 (at least 1, at most 4) NA       0.0219
## 39 male     Variable Matching 2 (at least 2, at most 4) NA       0.0487
## 40 male     Variable Matching 3 (Hansen's equation)     NA       0.00974
## 41 male     Variable Matching 4 (at least 2, at most 7) NA       0.0325
## 42 male     Pair matching                               NA       0.0873
```

Post-Full-Matching Analysis of Outcome

Hodges-Lehmann Aligned Rank Test The Hodges-Lehmann aligned rank test can be used on matched samples created by full matching or variable matching. We used full matching and found that children who used AFDC had a letter-word identification score in 1997 that was on average 1.97 points lower than those who had never used AFDC; the difference was statistically significant at a .05 level (one-tailed). We used the Hodges-Lehmann test to gauge the statistical significance. The study also detected an effect size of .19, which is a small effect size in terms of Cohen's (1988) criteria.

```
hodgesl(df, lwss97, fm, kuse)
```

```
## # A tibble: 1 x 6
##   HL_mean HL_se     z      p     i tx_effect
##     <dbl> <dbl> <dbl>  <dbl> <int>     <dbl>
## 1  -7039. 3247. -2.17 0.0151     1     -1.97
```

```
imbalance2(df, lwss97, kuse, fm)$dxm # dxm for the outcome variable is Cohen's d
```

Cohen's d

```
## [1] 0.1896308
```

Post-Matching Analysis Using Regression of Difference Scores

A regression of difference scores may be performed on matched samples created by optimal pair matching. Based on the pair-matched sample, we first calculated difference scores between treated and control cases for each pair on all study variables (i.e., on the outcome variable and all covariates). We then regressed the difference score of the outcome on the difference scores of covariates. In addition, note that our model includes a correction for clustering effects (i.e., children are nested within caregivers) that we accomplished by using robust estimates of standard errors. The intercept of a difference score regression indicates the ATE of the sample. The estimated intercept from this model is -3.17 ($p < .05$). Thus, using pair matching and regression adjustment, the study found that, on average, children who used AFDC had a letter-word identification score in 1997 that was 3.17 points lower than do children who never used AFDC; this finding was statistically significant.

```r
# kuse == 1
df1 <- df %>%
  select(pm, kuse,
    y1 = lwss97, male1 = male, black1 = black, age971 = age97,
    pcged971 = pcged97, mratio961 = mratio96, pcg_id
  ) %>%
  filter(kuse == 1)

# kuse == 0
df0 <- df %>%
  select(pm, kuse,
    y0 = lwss97, male0 = male, black0 = black, age970 = age97,
    pcged970 = pcged97, mratio960 = mratio96, pcg_id
  ) %>%
  filter(kuse == 0)

# Merge Data
df2 <- left_join(df0, df1, "pm")
df2 <- df2 %>%
  mutate(
    y = y1 - y0,
    male = male1 - male0,
    black = black1 - black0,
    age97 = age971 - age970,
    pcged97 = pcged971 - pcged970,
    mratio96 = mratio961 - mratio960
  )

# Regression of Difference Scores (one-tailed test)
reg1 <- lm(y ~ male + black + age97 + pcged97 + mratio96, data = df2)
broom::tidy(lmtest::coeftest(reg1, vcov = vcovCL, cluster = df2$pcg_id.x)) %>%
  mutate(p.value.one.tailed = p.value / 2)
```

```
## # A tibble: 6 x 6
##   term        estimate std.error statistic p.value p.value.one.tailed
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>              <dbl>
## 1 (Intercept)  -3.17       1.73    -1.83    0.0682              0.0341
## 2 male         -1.21       1.84    -0.657   0.512               0.256
## 3 black        -3.34       2.33    -1.43    0.153               0.0765
## 4 age97         0.229      0.329    0.695   0.487               0.244
## 5 pcged97       0.0111     0.508    0.0219  0.983               0.491
## 6 mratio96      3.15       1.67     1.89    0.0602              0.0301
```

Optimal (Pair) Matching Using the MatchIt Package

For an easier interface for the `optmatch` package, the `MatchIt::matchit()` function also implements optimal full matching and optimal pair matching with the `method = "full"` and `method = "optimal"` arguments, respectively. Variable matching can be set using the `ratio`, `min.controls`, and `max.controls` arguments.

Below is an illustration of optimal pair matching (by default, 1:1 without replacement) using GBM to generate the propensity scores and then a regression model for the outcome that incorporates the matching weights. Here we find that on average children who used AFDC had a letter-word identification score in 1997 that was 4.13 points lower than do children who never used AFDC ($p < .001$). However, post-matching balance checks revealed persistent covariate imbalances, therefore further analyses are warranted.

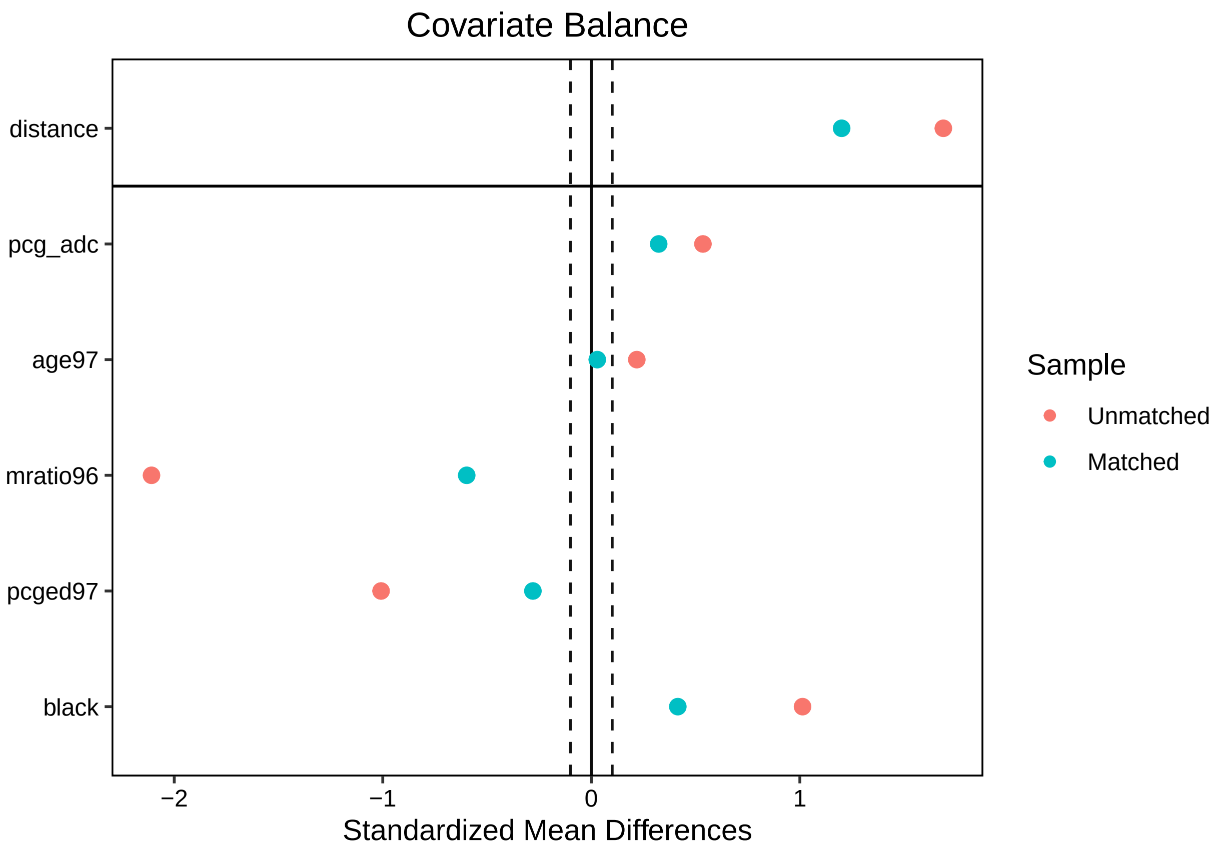
```r
# Optimal 1:1 Pair Matching Without Replacement Using GBM (ATT)
set.seed(1000)
(pm2 <- matchit(
  formula = kuse ~ pcg_adc + age97 + mratio96 + pcged97 + black,
  data = d,
  method = "optimal",
  replace = F,
  ratio = 1,
  distance = "gbm",
  estimand = "ATT"
))
```

```
## A ‘matchit‘ object
##  - method: 1:1 optimal pair matching
##  - distance: Propensity score
##              - estimated with GBM
##  - number of obs.: 1003 (original), 548 (matched)
##  - target estimand: ATT
##  - covariates: pcg_adc, age97, mratio96, pcged97, black
```

```r
# Balance Checks
cobalt::bal.tab(pm2, binary = "std", un = T, threshold = c(m = .1))
```

```
## Balance Measures
##              Type Diff.Un Diff.Adj        M.Threshold
## distance Distance  1.6880   1.2004
## pcg_adc    Contin.  0.5352   0.3230 Not Balanced, >0.1
## age97      Contin.  0.2181   0.0285     Balanced, <0.1
## mratio96   Contin. -2.1090  -0.5976 Not Balanced, >0.1
## pcged97    Contin. -1.0082  -0.2802 Not Balanced, >0.1
## black       Binary  1.0129   0.4148 Not Balanced, >0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         1
## Not Balanced, >0.1     4
##
## Variable with the greatest mean difference
##  Variable Diff.Adj        M.Threshold
##  mratio96  -0.5976 Not Balanced, >0.1
##
## Sample sizes
##           Control Treated
## All           729     274
## Matched       274     274
## Unmatched     455       0
```

```
cobalt::love.plot(pm2, un = T, binary = "std", threshold = c(m = .1),
                  sample.names = c("Unmatched", "Matched"))
```



```
# Get matched data
md <- match.data(pm2)

# Individual matches can be inspected using the "subclass" column
md %>% filter(subclass == 1)
```

```
## # A tibble: 2 x 16
##       kid  kuse   ps2  pcg_id age97 lwss97 k_adc pcged97 pcg_adc mratio96 black
##     <dbl> <dbl> <dbl>   <dbl> <dbl>  <dbl> <dbl>   <dbl>   <dbl>    <dbl> <dbl>
## 1 5600032     1 0.298 5600005     7    114  61.5      12       0     1.11     1
## 2 2277184     0 0.237 2277181     5     65    0         12       2     1.60     0
## # i 5 more variables: male <dbl>, puse <dbl+lbl>, distance <dbl>,
## #   weights <dbl>, subclass <fct>
```

```
# Outcome models with matching weights and cluster-robust SEs
pm2.lm1 <- lm(lwss97 ~ kuse, data = md, weights = weights)
pm2.lm2 <- lm(lwss97 ~ kuse * (male + black + age97 + pcged97 + mratio96), data = md,
             weights = weights)
pm2.lm3 <- lm(lwss97 ~ kuse + male + black + age97 + pcged97 + mratio96 + distance,
              data = md, weights = weights)
pm2.lm4 <- lm(lwss97 ~ kuse + male + black + age97 + pcged97 + mratio96, data = md,
             weights = weights) # doubly-robust
broom::tidy(lmtest::coeftest(pm2.lm1, vcov = vcovCL, cluster = md$pcg_id))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   100.        1.09     91.5  0
## 2 kuse           -5.84      1.47     -3.98 0.0000774
```

```
broom::tidy(lmtest::coeftest(pm2.lm2, vcov = vcovCL, cluster = md$pcg_id))
```

```
## # A tibble: 12 x 5
##    term           estimate std.error statistic  p.value
##    <chr>             <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     81.5       8.55      9.54  5.01e-20
##  2 kuse             3.68     11.5       0.321 7.48e- 1
##  3 male            -4.47      1.87     -2.39  1.72e- 2
##  4 black           -1.72      2.36     -0.731 4.65e- 1
##  5 age97            0.792     0.335     2.37  1.82e- 2
##  6 pcged97          0.977     0.703     1.39  1.65e- 1
##  7 mratio96         2.83      1.05      2.71  7.04e- 3
##  8 kuse:male        3.96      2.66      1.49  1.37e- 1
##  9 kuse:black      -1.02      3.44     -0.296 7.68e- 1
## 10 kuse:age97      -0.475     0.458    -1.04  3.00e- 1
## 11 kuse:pcged97    -0.343     0.937    -0.366 7.15e- 1
## 12 kuse:mratio96   -1.18      1.44     -0.819 4.13e- 1
```

```
broom::tidy(lmtest::coeftest(pm2.lm3, vcov = vcovCL, cluster = md$pcg_id))
```

```
## # A tibble: 8 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   89.5       7.00     12.8   7.32e-33
## 2 kuse          -3.11      1.90     -1.64  1.02e- 1
## 3 male          -2.55      1.29     -1.98  4.86e- 2
## 4 black         -1.21      1.89     -0.638 5.24e- 1
## 5 age97          0.576     0.229     2.51  1.23e- 2
## 6 pcged97        0.595     0.484     1.23  2.19e- 1
## 7 mratio96       1.59      1.10      1.45  1.46e- 1
## 8 distance      -4.15      4.85     -0.856 3.92e- 1
```

```r
broom::tidy(lmtest::coeftest(pm2.lm4, vcov = vcovCL, cluster = md$pcg_id))
```

```
## # A tibble: 7 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   85.6       5.83      14.7  2.35e-41
## 2 kuse          -3.98      1.53      -2.60 9.68e- 3
## 3 male          -2.56      1.29      -1.99 4.73e- 2
## 4 black         -1.90      1.72      -1.11 2.69e- 1
## 5 age97          0.563     0.228      2.47 1.37e- 2
## 6 pcged97        0.778     0.463      1.68 9.39e- 2
## 7 mratio96       2.23      0.717      3.12 1.93e- 3
```

```r
# Estimate ATT with marginaleffects::avg_comparisons() (G-computation)
marginaleffects::avg_comparisons(pm2.lm4,
  variables = "kuse", # treatment
  vcov = ~subclass + pcg_id, # cluster-robust SEs
  newdata = subset(md, kuse == 1), # ATT
  wts = "weights"
)
```

```
##
##  Estimate Std. Error     z Pr(>|z|)   S 2.5 % 97.5 %
##     -3.98       1.55 -2.57   0.0101 6.6 -7.01 -0.949
##
## Term: kuse
## Type: response
## Comparison: 1 - 0
```

5.5 Computer Lab: Running the IPTW Estimator with R

Load Packages

Propensity score weighting can be accomplished with base R functions. However, we need the `lmtest` and `sandwich` packages to estimate clustered covariance matrices in this example. Using these packages, we can obtain estimates and standard errors that are identical to Stata's `regress` program.

```r
pacman::p_load(lmtest, sandwich)
```

Description of Dataset

This dataset is from a study that investigates intergenerational dependence on welfare and its relation to child academic achievement.[16]

The dependent variable is `lwss97` or "letter-word identification" score, and the treatment condition is `kuse` or children who used Aid to Families With Dependent Children (AFDC). The covariates are:

- `male`: Child's Gender: Male (Reference: Female)
- `black`: Child's Race: African American (Reference: Other)
- `age97`: Child's Age in 1997
- `pcged97`: Caregiver's Education in 1997 (Years of Schooling)
- `mratio96`: Ratio of Family Income to Poverty Line in 1996

Additionally, `pcg_id` is a cluster variable that identifies children nested within families.

Estimate ATE and ATT Weights

Separate weights need to be calculated for estimating the average treatment effect (ATE) and the average treatment effect for the treated (ATT). Propensity score weighting for estimating ATE is generally referred to as the inverse probability of treatment weights (IPTW) estimator.

For ATE, the weight estimates are calculated as follows for the treatment group:

$$\omega = \frac{1}{\hat{e}(x)}$$

And for the control group:

$$\omega = \frac{1}{1 - \hat{e}(x)}$$

For ATT, the weight is 1 for a treated case. The weight for a comparison case is:

$$\omega = \frac{\hat{e}(x)}{1 - \hat{e}(x)}$$

[16] Hofferth, S., Stafford, F. P., Yeung, W. J., Duncan, G. J., Hill, M. S., Lepkowski, J., et al. (2001). *Panel study of income dynamics, 1968–1999: Supplemental files (computer file), ICPSR version.* Ann Arbor: University of Michigan Survey Research Center.

Load Data with Propensity Scores and Calculate Weights

```
psw_df <- read_dta("data/chpt5_2_original.dta") %>%
  haven::zap_formats() %>%
  sjlabelled::remove_all_labels() %>%
  as_tibble() %>%
  mutate(
    ate_w = ifelse(kuse == 0, 1 / (1 - ps), 1 / ps),
    att_w = ifelse(kuse == 0, ps / (1 - ps), 1)
  )
```

Check Balance

In addition to standardized mean differences, variance ratios can be requested. Common thresholds for balanced groups are .5 and 2, but it will be close to 1 when group variances are similar. Remember to use 's.d.denom = "pooled"' for the ATE and 's.d.denom = "treated"' for the ATT.

```
# ATE
cobalt::bal.tab(
  x = select(psw_df, male, black, age97, pcged97, mratio96),
  treat = psw_df$kuse,
  binary = "std",
  continuous = "std",
  s.d.denom = "pooled",
  un = T,
  stats = c("mean.diffs", "variance.ratios"),
  thresholds = c(m = .1, v = 2)
)
```

```
## Balance Measures
##             Type Diff.Un     M.Threshold.Un V.Ratio.Un   V.Threshold.Un
## male      Binary -0.0335     Balanced, <0.1          .
## black     Binary  0.9343 Not Balanced, >0.1          .
## age97    Contin.  0.2196 Not Balanced, >0.1     1.0266     Balanced, <2
## pcged97  Contin. -0.9067 Not Balanced, >0.1     0.6789     Balanced, <2
## mratio96 Contin. -1.0158 Not Balanced, >0.1     0.1312 Not Balanced, >2
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         1
## Not Balanced, >0.1     4
##
## Variable with the greatest mean difference
##  Variable Diff.Un     M.Threshold.Un
##  mratio96 -1.0158 Not Balanced, >0.1
##
## Balance tally for variance ratios
##                  count
## Balanced, <2         2
## Not Balanced, >2     1
##
## Variable with the greatest variance ratio
##  Variable V.Ratio.Un   V.Threshold.Un
##  mratio96     0.1312 Not Balanced, >2
##
## Sample sizes
##     Control Treated
## All     729     274
```

```
# ATT
cobalt::bal.tab(
  x = select(psw_df, male, black, age97, pcged97, mratio96),
  treat = psw_df$kuse,
  binary = "std",
  continuous = "std",
  s.d.denom = "treated",
  un = T,
  stats = c("mean.diffs", "variance.ratios"),
  thresholds = c(m = .1, v = 2)
)
```

```
## Balance Measures
##             Type Diff.Un     M.Threshold.Un V.Ratio.Un   V.Threshold.Un
## male      Binary -0.0335     Balanced, <0.1          .
## black     Binary  1.0129 Not Balanced, >0.1          .
## age97    Contin.  0.2181 Not Balanced, >0.1     1.0266     Balanced, <2
## pcged97  Contin. -1.0082 Not Balanced, >0.1     0.6789     Balanced, <2
## mratio96 Contin. -2.1090 Not Balanced, >0.1     0.1312 Not Balanced, >2
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         1
## Not Balanced, >0.1     4
##
## Variable with the greatest mean difference
##  Variable Diff.Un     M.Threshold.Un
##  mratio96  -2.109 Not Balanced, >0.1
##
## Balance tally for variance ratios
##                  count
## Balanced, <2         2
## Not Balanced, >2     1
##
## Variable with the greatest variance ratio
##  Variable V.Ratio.Un   V.Threshold.Un
##  mratio96     0.1312 Not Balanced, >2
##
## Sample sizes
##     Control Treated
## All     729     274
```

Calculate Weights with the WeightIt Package

```r
# Load Package
pacman::p_load(WeightIt)

# Estimate ATE and ATT weights and Compare with Previous Results
ate_w2 <- WeightIt::get_w_from_ps(ps = psw_df$ps, treat = psw_df$kuse, estimand = "ATE")
table(ate_w2 == psw_df$ate_w)
```

```
##
## TRUE
## 1003
```

```r
att_w2 <- WeightIt::get_w_from_ps(ps = psw_df$ps, treat = psw_df$kuse, estimand = "ATT")
table(att_w2 == psw_df$att_w)
```

```
##
## TRUE
## 1003
```

Outcome Analysis

Weighted Regression with ATE Weights

After creating the weights, use the `weights` argument in `lm()` to run a weighted outcome analysis and `lmtest::coeftest()` to control for clustering effects.

This analysis showed that children who used Aid to Families With Dependent Children (AFDC) had an average letter-word identification score that was 5.16 points lower than children who never used AFDC, $p < .01$.

```r
# ATE
psw_ate <- lm(lwss97 ~ kuse + male + black + age97 + pcged97 + mratio96,
  data = psw_df, weights = ate_w
)
lmtest::coeftest(psw_ate, vcov. = vcovCL(psw_ate, cluster = psw_df$pcg_id))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value              Pr(>|t|)
## (Intercept) 84.19992    4.83032 17.4315 < 0.00000000000000022 ***
## kuse        -5.16399    1.42438 -3.6254             0.0003031 ***
## male        -1.62201    1.09186 -1.4855             0.1377180
## black       -2.49898    1.34670 -1.8556             0.0638009 .
## age97        0.73868    0.18075  4.0867            0.00004727 ***
## pcged97      0.99264    0.35596  2.7886             0.0053938 **
## mratio96     1.13856    0.32220  3.5337             0.0004286 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Weighted Regression with ATT Weights

When considering only individuals assigned to the treatment condition, children who used AFDC had an average letter-word identification score that was 4.62 points lower than children who never used AFDC, $p < .01$.

```r
# ATT
psw_att <- lm(lwss97 ~ kuse + male + black + age97 + pcged97 + mratio96,
  data = psw_df, weights = att_w
)
lmtest::coeftest(psw_att, vcov. = vcovCL(psw_att, cluster = psw_df$pcg_id))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value              Pr(>|t|)
## (Intercept) 85.29467    5.05331 16.8790 < 0.00000000000000022 ***
## kuse        -4.62058    1.41182 -3.2728              0.001102 **
## male        -1.58995    1.14705 -1.3861              0.166020
## black       -2.74605    1.41605 -1.9392              0.052756 .
## age97        0.61577    0.20001  3.0787              0.002136 **
## pcged97      0.92698    0.36718  2.5246              0.011738 *
## mratio96     1.26018    0.33556  3.7555              0.000183 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Check Balance

To assess balance before and after propensity score weighting, we can use weighted logistic regression for the dummy covariates and weighted simple regression for the continuous covariates. Some examples are included below, and the full code can be found in Section 7.3.1 of the PSA-R code.

In model `psw_c3` below, the treatment dummy variable is significant, meaning that there is no sufficient balance after the propensity score weighting.

To assess balance before propensity score weighting, remove the `weights` argument.

```r
psw_c1 <- glm(male ~ kuse, family = quasibinomial, data = psw_df, weights = ate_w)
lmtest::coeftest(psw_c1, vcov. = vcovCL(psw_c1, cluster = psw_df$pcg_id))
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.142911   0.075538  1.8919   0.0585 .
## kuse        -0.060271   0.150143 -0.4014   0.6881
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
psw_c2 <- glm(male ~ kuse, family = quasibinomial, data = psw_df, weights = att_w)
lmtest::coeftest(psw_c2, vcov. = vcovCL(psw_c2, cluster = psw_df$pcg_id))
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.152522   0.077410  1.9703   0.0488 *
## kuse        -0.079497   0.147480 -0.5390   0.5899
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
psw_c3 <- lm(age97 ~ kuse, weights = ate_w, data = psw_df)
lmtest::coeftest(psw_c3, vcov. = vcovCL(psw_c3, cluster = psw_df$pcg_id))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value              Pr(>|t|)
## (Intercept)  6.51389    0.10900 59.7615 < 0.00000000000000022 ***
## kuse         0.61064    0.21883  2.7905              0.005362 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
psw_c4 <- lm(age97 ~ kuse, weights = att_w, data = psw_df)
lmtest::coeftest(psw_c4, vcov. = vcovCL(psw_c4, cluster = psw_df$pcg_id))
```

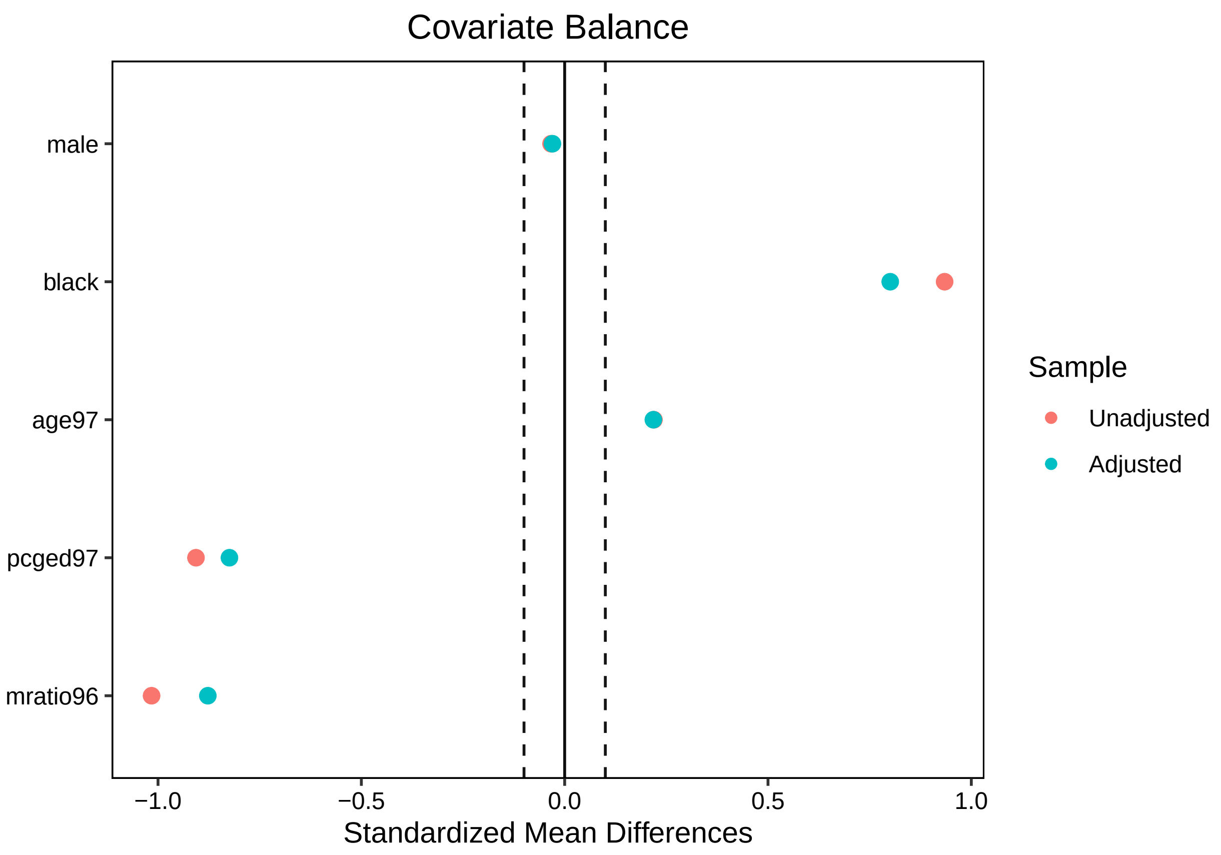
```
##
## t test of coefficients:
##
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  6.55501    0.11499 57.0057 < 0.0000000000000002 ***
## kuse         0.56178    0.21939  2.5606              0.01059 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alternatively, balance can be assessed using the `cobalt` package.

```
# ATE
(psw_b1 <- cobalt::bal.tab(
  x = select(psw_df, male, black, age97, pcged97, mratio96),
  treat = psw_df$kuse,
  weights = psw_df$ate_w,
  binary = "std",
  continuous = "std",
  s.d.denom = "pooled",
  un = T,
  stats = c("mean.diffs", "variance.ratios"),
  thresholds = c(m = .1, v = 2)
))
```

```
## Balance Measures
##             Type Diff.Un V.Ratio.Un Diff.Adj        M.Threshold V.Ratio.Adj
## male      Binary -0.0335          .  -0.0301     Balanced, <0.1           .
## black     Binary  0.9343          .   0.8006 Not Balanced, >0.1           .
## age97    Contin.  0.2196     1.0266   0.2181 Not Balanced, >0.1      1.0089
## pcged97  Contin. -0.9067     0.6789  -0.8244 Not Balanced, >0.1      0.6940
## mratio96 Contin. -1.0158     0.1312  -0.8775 Not Balanced, >0.1      0.1781
##               V.Threshold
## male
## black
## age97        Balanced, <2
## pcged97      Balanced, <2
## mratio96 Not Balanced, >2
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         1
## Not Balanced, >0.1     4
##
## Variable with the greatest mean difference
##  Variable Diff.Adj        M.Threshold
##  mratio96  -0.8775 Not Balanced, >0.1
##
## Balance tally for variance ratios
##                  count
## Balanced, <2         2
## Not Balanced, >2     1
##
## Variable with the greatest variance ratio
##  Variable V.Ratio.Adj      V.Threshold
##  mratio96      0.1781 Not Balanced, >2
##
## Effective sample sizes
##            Control Treated
## Unadjusted  729.    274.
## Adjusted    725.43  257.71
```

```
cobalt::love.plot(psw_b1)
```

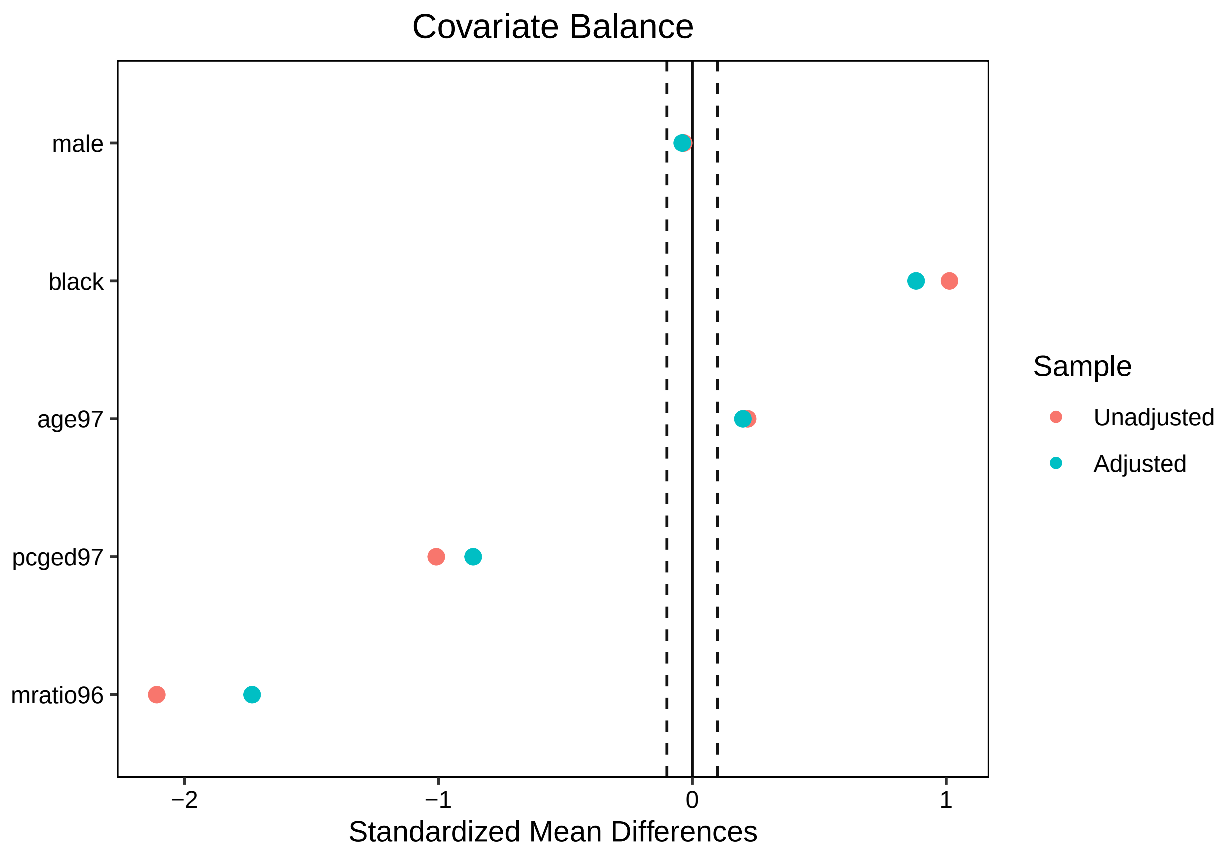


```
# ATT
(psw_b2 <- cobalt::bal.tab(
  x = select(psw_df, male, black, age97, pcged97, mratio96),
  treat = psw_df$kuse,
  weights = psw_df$att_w,
  binary = "std",
  continuous = "std",
  s.d.denom = "treated",
  un = T,
  stats = c("mean.diffs", "variance.ratios"),
  thresholds = c(m = .1, v = 2)
))
```

```
## Balance Measures
##             Type Diff.Un V.Ratio.Un Diff.Adj        M.Threshold V.Ratio.Adj
## male      Binary -0.0335          .  -0.0396     Balanced, <0.1           .
## black     Binary  1.0129          .   0.8814 Not Balanced, >0.1           .
## age97    Contin.  0.2181     1.0266   0.1993 Not Balanced, >0.1      1.0085
## pcged97  Contin. -1.0082     0.6789  -0.8629 Not Balanced, >0.1      0.6694
## mratio96 Contin. -2.1090     0.1312  -1.7335 Not Balanced, >0.1      0.1463
##               V.Threshold
## male
## black
## age97        Balanced, <2
## pcged97      Balanced, <2
## mratio96 Not Balanced, >2
```

```
##
## Balance tally for mean differences
##                     count
## Balanced, <0.1          1
## Not Balanced, >0.1      4
##
## Variable with the greatest mean difference
##  Variable Diff.Adj        M.Threshold
##  mratio96  -1.7335 Not Balanced, >0.1
##
## Balance tally for variance ratios
##                   count
## Balanced, <2          2
## Not Balanced, >2      1
##
## Variable with the greatest variance ratio
##  Variable V.Ratio.Adj      V.Threshold
##  mratio96      0.1463 Not Balanced, >2
##
## Effective sample sizes
##            Control Treated
## Unadjusted  729.       274
## Adjusted    663.65     274
```

```r
cobalt::love.plot(psw_b2)
```



Using WeightIt

The WeightIt package can generate propensity scores and the appropriate weights in a single step.

```r
# Estimate propensity scores and generate ATT weights
psw_g1 <- WeightIt::weightit(
  kuse ~ male + black + age97 + pcged97 + mratio96,
  data = psw_df, estimand = "ATT", method = "glm"
)
```

```
## Warning: Propensity scores numerically equal to 0 or 1 were estimated,
## indicating perfect separation and infinite parameter estimates. These
## may yield problems with inference. Consider trying a different ‘link‘.
## See ‘help("method_glm", package = "WeightIt")‘ for details.
```

```r
# Check balance
cobalt::bal.tab(
  psw_g1,
  binary = "std",
  continuous = "std",
  s.d.denom = "treated", # att
  un = T,
  stats = c("m"),
  thresholds = c(m = .1)
)
```

```
## Balance Measures
##                Type Diff.Un Diff.Adj        M.Threshold
## prop.score Distance  1.5861   0.1758
## male         Binary -0.0335  -0.0208     Balanced, <0.1
## black        Binary  1.0129   0.0628     Balanced, <0.1
## age97       Contin.  0.2181   0.0363     Balanced, <0.1
## pcged97     Contin. -1.0082   0.0587     Balanced, <0.1
## mratio96    Contin. -2.1090  -0.1460 Not Balanced, >0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         4
## Not Balanced, >0.1     1
##
## Variable with the greatest mean difference
##  Variable Diff.Adj        M.Threshold
##  mratio96   -0.146 Not Balanced, >0.1
##
## Effective sample sizes
##            Control Treated
## Unadjusted   729.      274
## Adjusted     141.1     274
```

```r
# Attach ATT weights to dataframe
psw_df2 <- psw_df %>% mutate(att_weights = psw_g1$weights)

# Fit outcome model and estimate treatment effect with cluster-robust SEs
```

```r
psw_g2 <- lm(lwss97 ~ kuse + male + black + age97 + pcged97 + mratio96,
  data = psw_df2, weights = att_weights
)
marginaleffects::avg_comparisons(
  psw_g2,
  variables = "kuse",
  vcov = ~pcg_id,
  newdata = subset(psw_df2, kuse == 1), # att
  wts = "att_weights"
)
```

```
##
##  Estimate Std. Error     z Pr(>|z|)   S 2.5 % 97.5 %
##     -3.94       1.67 -2.36   0.0182 5.8  -7.2 -0.671
##
## Term: kuse
## Type: response
## Comparison: 1 - 0
```

```r
# How would you calculate the ATE weights using this approach?
```

6.9 Computer Lab: Running the Matching Estimators with R

Matching Estimators

Load Packages

A variety of matching estimators are implemented in the `Matching` package. Unlike the `MatchIt` package, which uses "subset selection" to arrive at a weighted subset of the units from the dataset, the `Matching` package uses "matching imputation" to impute potential outcomes using the observed outcomes of paired units.[17]

Because the assumptions about constant treatment effect and homoskedasticity may not be valid for certain types of data, we will use the `lmtest` package for the Breusch-Pagan Test to check this assumption.

```r
pacman::p_load(Matching)
```

Note that the `Matching::Match()` function is intended to be used in conjunction with the `MatchBalance()` function. However, functions from the `cobalt` package also work well and tend to be cleaner in presentation:[18]

```r
data(lalonde, package = "Matching")
ex_f <- as.formula(treat ~ age + I(age^2) + educ + I(educ^2) + black +
  hisp + married + nodegr + re74 + I(re74^2) + re75 + I(re75^2) +
  u74 + u75)
ex_m1 <- glm(ex_f, family = binomial, data = lalonde)
ex_p <- ex_m1$fitted.values
ex_X <- ex_m1$fitted
ex_Y <- lalonde$re78
ex_Tr <- lalonde$treat
ex_rr <- Match(Y = ex_Y, Tr = ex_Tr, X = ex_X, M = 1)
summary(ex_rr)
ex_mb <- MatchBalance(treat ~ age + I(age^2) + educ + I(educ^2) + black +
  hisp + married + nodegr + re74 + I(re74^2) + re75 + I(re75^2) +
  u74 + u75, data = lalonde, match.out = ex_rr, nboots = 10)
(ex_bal <- cobalt::bal.tab(ex_rr, ex_f,
  data = lalonde, distance = ~ex_p, un = T,
  binary = "std", threshold = .1
))
cobalt::love.plot(ex_bal, sample.names = c("Unmatched", "Matched"))
```

Description of Dataset

This example uses the 1997 Child Development Supplement (CDS) to the Panel Study of Income Dynamics (PSID) and the core PSID annual data from 1968 to 1997.

The dependent variable in this dataset is `pcss97`, a passage comprehension score. Higher scores on this measure indicate higher academic achievement. The treatment variable is `kuse` or children who ever used Aid to Families With Dependent Children (AFDC). The covariates or matching variables are:

- `pcg_adc`: Caregiver's History of Using Welfare (Number of Years; range: 0-7)
- `age97`: Child's Age in 1997

[17] https://kosukeimai.github.io/MatchIt/articles/matching-methods.html
[18] https://ngreifer.github.io/cobalt/articles/other-packages.html

- `mratio96`: Ratio of Family Income to Poverty Line in 1996
- `pcged97`: Caregiver's Education in 1997 (Years of Schooling)
- `male`: Child's Gender: Male (1 = Male; 0 = Female)
- `black`: Child's Race: African American (1 = African American; 0 = Other)

Load Data

```
me_df <- haven::read_dta("data/cds_pcss97.dta") %>%
  haven::zap_formats() %>%
  sjlabelled::remove_all_labels() %>%
  as_tibble()
head(me_df) %>%
  kbl(booktabs = T, linesep = "", digits = 2) %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

kid	pcg_id	age97	pcss97	k_adc	pcged97	pcg_adc	mratio96	black	male	kuse
4180	4179	12	93	7.69	9	0	0.41	0	0	1
5032	5170	12	122	0.00	12	0	5.00	0	0	0
7041	7030	6	93	0.00	11	2	1.07	0	1	0
10033	10177	7	101	8.33	12	0	2.40	1	0	1
10034	10006	6	127	0.00	14	0	1.00	1	0	0
14030	14172	11	103	0.00	12	0	3.77	0	1	0

Breusch-Pagan Test for Heteroskedasticity

The homoskedastic variance estimator assumes that the unit-level treatment effect is constant and that the conditional variance of $Y_i(w)$ given X_i does not vary with either covariates or the treatment.

To carry out the Breusch-Pagan Test, first we regress the outcome variable on the matching variables using OLS:

```
# Regress outcome on treatment and matching variables using OLS
me_m1 <- lm(pcss97 ~ kuse + male + black + age97 + pcged97 + mratio96 + pcg_adc,
  data = me_df
)
```

Next, we can run the Breusch-Pagan test for each matching variable:

```
lmtest::bptest(me_m1, ~kuse, data = me_df, studentize = F)
lmtest::bptest(me_m1, ~male, data = me_df, studentize = F)
lmtest::bptest(me_m1, ~black, data = me_df, studentize = F)
lmtest::bptest(me_m1, ~age97, data = me_df, studentize = F) # significant
lmtest::bptest(me_m1, ~pcged97, data = me_df, studentize = F)
lmtest::bptest(me_m1, ~mratio96, data = me_df, studentize = F)
lmtest::bptest(me_m1, ~pcg_adc, data = me_df, studentize = F)
```

Or use a function to test every variable:

```r
bp <- function(var, df, md) {
  lmtest::bptest(md, as.formula(paste0("~", var)), data = df, studentize = F) %>%
    broom::tidy() %>%
    mutate(variable = var) %>%
    select(variable, statistic, p.value)
}
map_dfr(c("kuse", "male", "black", "age97", "pcged97", "mratio96", "pcg_adc"), bp,
  df = me_df, md = me_m1
) %>%
  kbl(
    booktabs = T, linesep = "", digits = 2,
    caption = "Results of Breusch-Pagan Tests for Heteroskedasticity"
  ) %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 1: Results of Breusch-Pagan Tests for Heteroskedasticity

variable	statistic	p.value
kuse	0.01	0.92
male	3.07	0.08
black	1.43	0.23
age97	25.27	0.00
pcged97	0.19	0.66
mratio96	0.62	0.43
pcg_adc	0.29	0.59

Results from the Breusch-Pagan tests showed that the homoskedasticity assumption is not valid for child's age (`age97`) ($p < .05$) and indicated that the conditional variance of the outcome variable was not constant across levels of child's age. Based on this finding, we should use the robust variance estimator that allows for heteroskedasticity (i.e., the `Var.calc` argument in the `Matching::Match()` function).

Matching Estimators

Of the six matching variables in this example, four are continuous and two are categorical, therefore bias-corrected matching estimator is necessary to correct for bias corresponding to the matching discrepancies between matched units and their matches on the four continuous covariates. Regression adjustment can be used with the `BiasAdjust = T` argument. (Tip: When you have one or more continuous covariate in your matching, always use the bias-corrected matching estimator.)

By default, the `Matching::Match()` function uses the matching variables to make bias adjustments. However, these covariates can be specified using the `Z` argument (example shown below).

The `M` argument specifies the number of matches which should be found. The default is one-to-one matching (i.e., M = 1). Abadie and Imbens suggest that M be small, and `M = 4` typically performs well in terms of mean-squared error.

If `Var.calc = 0`, then homoskedasticity is assumed. Use `Var.calc = 4` to request the robust variance estimator using four matches. This algorithm developed by Abadie and Imbens (2002) includes a second matching procedure such that it matches treated units to treated units and control units to controls.

The `estimand` argument is by default "ATT" but can be set to "ATE" or "ATC". Typically, we are interested in the ATE or ATT. The ATT seeks to answer questions such as "How would treated patients' outcomes differ, on average, had they not received treatment?"; the ATE is the treatment effect for both control and treated units; and the ATC, the average treatment effect for the controls, investigates the effect of the treatment for a population who did not receive treatment.[19]

The `sample` argument is a logical flag indicating whether the population or sample variance should be estimated. An example may help illustrate the difference between PATE and SATE: "While the SATE is useful for judging how a job-training program has affected a particular group of participants, the PATE can be used to evaluate whether another group of participants drawn from the same population is likely to benefit from the program."[20] In other words, the sample effect shows whether the program is successful in the sample at hand, while the population effect shows whether the same program would be successful in a second sample from the population.

Results from the `Matching::Match()` function are identical to Stata's `nnmatch` program.

Define Outcome (Y), Treatment Index (Tr), and Variables to Match On (X)

Use the `Y`, `Tr`, and `X`, arguments in the `Match()` function to specify the outcome vector, the treatment vector, and the matrix of variables to match on, respectively. The X matrix may or may not contain the propensity score.

```r
me_Y <- me_df$pcss97
me_Tr <- me_df$kuse
me_X <- select(me_df, male, black, age97, pcged97, mratio96, pcg_adc)
```

Get Estimators Individually

Note that by default matching is done with replacement. However, this can be changed with the `replace` argument. A matched dataset can be recovered by using the `index.treated` and `index.control` vectors. Further, `index.dropped` contains observations that have been dropped (e.g., due to a caliper setting).

```r
# Sample Average Treatment Effect (SATE)
me1 <- Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 4, BiasAdjust = T, Var.calc = 4,
  estimand = "ATE", sample = T
)
summary(me1)
head(me1$index.treated)
head(me1$index.control)
head(me1$index.dropped)
head(me1$se) # Abadie-Imbens standard error

# Population Average Treatment Effect (PATE)
summary(Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 4, BiasAdjust = T, Var.calc = 4,
  estimand = "ATE", sample = F
))

# Sample average treatment effect for the treated (SATT)
summary(Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 4, BiasAdjust = T, Var.calc = 4,
  estimand = "ATT", sample = T
```

[19]https://arxiv.org/abs/2106.10577
[20]https://journals.sagepub.com/doi/pdf/10.1177/1536867X0400400307

```r
))

# Population average treatment effect for the treated (PATT)
summary(Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 4, BiasAdjust = T, Var.calc = 4,
  estimand = "ATT", sample = F
))

# Sample average treatment effect for the controls (SATC)
summary(Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 4, BiasAdjust = T, Var.calc = 4,
  estimand = "ATC", sample = T
))

# Population average treatment effect for the controls (PATC)
summary(Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 4, BiasAdjust = T, Var.calc = 4,
  estimand = "ATC", sample = F
))
```

Get All Estimators

```r
# Function for extracting estimate, SE, t-stat, and p-value from Match()
get_match <- function(estimand, sample, Y, Tr, X) {
  m <- Matching::Match(
    Y = Y, Tr = Tr, X = X, M = 4, BiasAdjust = T, Var.calc = 4,
    estimand = estimand, sample = sample
  )
  return(list(
    est = m$est[, 1],
    se = m$se,
    t.stat = m$est[, 1] / m$se,
    p = (1 - pnorm(abs(m$est[, 1] / m$se))) * 2
  ))
}

# Estimate different matching estimators
tribble(
  ~estimator, ~estimand, ~sample,
  "SATE", "ATE", T,
  "PATE", "ATE", F,
  "SATT", "ATT", T,
  "PATT", "ATT", F,
  "SATC", "ATC", T,
  "PATC", "ATC", F
) %>%
  rowwise() %>%
  mutate(match = list(get_match(estimand, sample, me_Y, me_Tr, me_X))) %>%
  unnest_wider(match) %>%
  select(-estimand, -sample) %>%
  kbl(
    booktabs = T, linesep = "",
    caption = "Bias-Corrected Matching Estimators with Robust Standard Errors"
  ) %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 2: Bias-Corrected Matching Estimators with Robust Standard Errors

estimator	est	se	t.stat	p
SATE	-4.703773	1.769696	-2.657956	0.0078616
PATE	-4.703773	1.765187	-2.664746	0.0077047
SATT	-5.229651	1.781217	-2.935999	0.0033248
PATT	-5.229651	1.720590	-3.039451	0.0023701
SATC	-4.467254	2.133536	-2.093827	0.0362754
PATC	-4.467254	2.135647	-2.091757	0.0364602

The results suggest that childhood poverty strongly affected children's academic achievement.

ATE: On average, children who used AFDC in childhood had a passage comprehension score 4.7 units lower than that of children who had never used AFDC in childhood. This effect is statistically significant in the sample at hand (SATE $p < .05$) as well as in a second sample drawn from the same population (PATE $p < .05$).

ATT: With regard to the subpopulation of treated participants, on average, children who used AFDC in childhood had a passage comprehension score 5.2 units lower than that of children who had never used AFDC in childhood. This effect is statistically significant in the sample at hand (SATT $p < .05$) as well as in a second sample drawn from the same population (PATT $p < .05$).

ATC: Had all controls (i.e., children who never used AFDC) used AFDC and all treated children had not used AFDC, then on average, the control children would have a passage comprehension score 4.5 units lower than their counterparts (SATC $p < .05$; PATC $p < .05$).

Additional observations:

1. A population effect indicates whether the tested intervention will be effective in a second sample taken from the same population. Taking SATT and PATT as examples, the study indicated that the treatment effect for the treated group was statistically significant in the sample at a level of .01. If we take a second sample from the population, we are likely to observe the same level of treatment effect for the treated, and the effect should remain statistically significant at a level of .01. The point estimate of the population effect is identical to the point estimate of its corresponding sample effect. A population effect differs from its corresponding sample effect on variance, and thus significance test on a population effect may have a different conclusion than that on its corresponding sample effect. Note that if treated units are discarded (e.g., due to a caliper), the estimand no longer estimates a particular population but rather only the average treatment effect on the remaining matched units (ATM).

2. Note that in this study, SATT = -5.23 and SATC = -4.47, or a difference of 0.76 units. This difference is attributable either to additional selection bias that was not accounted for in the study or to study data that violated assumptions of matching estimators, which suggests the need for further scrutiny.

3. All six treatment effects were statistically significant ($p < .05$). Thus, we can conclude that the study data could not reject a null hypothesis of a zero treatment effect, and childhood poverty appears to be an important factor causing children's poor achievement in passage comprehension.

Specify Variables in the Bias-Corrected Matching

The Z argument can be used to specify the covariates for which we wish to make bias adjustments.

```
# Sample Average Treatment Effect (SATE)
me_Z <- select(me_df, age97, pcged97, mratio96, pcg_adc)
summary(Matching::Match(
  Y = me_Y, Tr = me_Tr, X = me_X, Z = me_Z, M = 4,
  BiasAdjust = T, Var.calc = 4, estimand = "ATE", sample = T
))
```

```
##
## Estimate...  -4.4867
## AI SE......  1.7697
## T-stat.....  -2.5353
## p.val......  0.011235
##
## Original number of observations..............  606
## Original number of treated obs...............  188
## Matched number of observations...............  606
## Matched number of observations  (unweighted).  2441
```

Check Balance

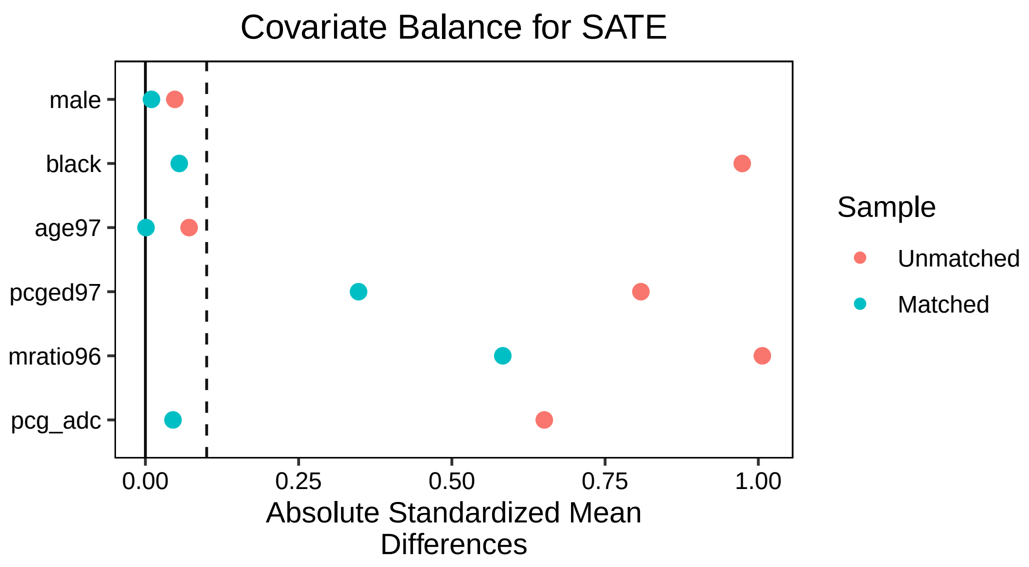
The `Matching::Match()` function works well in conjunction with the `cobalt::bal.tab()` function for checking covariate balance.

By default, the denominator for standardized mean differences uses a pooled estimate (square root of the average of the group variances) for ATE and the standard deviation of the treated group for ATT, and both standard deviations are computed using the sample before matching. This option can also be manually set with the `s.d.denom` option.

```
# Example of Checking Balance for SATE
me_SATE <- Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 4, BiasAdjust = T, Var.calc = 4,
  estimand = "ATE", sample = T
)
(me_SATE_bal <- cobalt::bal.tab(
  me_SATE, kuse ~ male + black + age97 + pcged97 + mratio96 + pcg_adc,
  data = me_df,
  abs = T,
  un = T,
  binary = "std",
  thresholds = c(m = .1),
  s.d.denom = "pooled" # ATE
))
```

```
## Balance Measures
##             Type Diff.Un Diff.Adj        M.Threshold
## male      Binary  0.0480   0.0099     Balanced, <0.1
## black     Binary  0.9738   0.0553     Balanced, <0.1
## age97    Contin.  0.0713   0.0010     Balanced, <0.1
## pcged97  Contin.  0.8084   0.3478 Not Balanced, >0.1
## mratio96 Contin.  1.0066   0.5831 Not Balanced, >0.1
## pcg_adc  Contin.  0.6508   0.0450     Balanced, <0.1
##
## Balance tally for mean differences
##                    count
## Balanced, <0.1         4
## Not Balanced, >0.1     2
##
## Variable with the greatest mean difference
##  Variable Diff.Adj        M.Threshold
##  mratio96   0.5831 Not Balanced, >0.1
##
## Sample sizes
##                      Control Treated
## All                   418.    188.
## Matched (ESS)         318.46   85.38
## Matched (Unweighted)  418.    188.
```

```
cobalt::love.plot(me_SATE_bal, sample.names = c("Unmatched", "Matched")) +
  labs(title = "Covariate Balance for SATE")
```



Outcome Analysis

```
# Recovering a matched dataset
me_out <- Match(
  Y = me_Y, Tr = me_Tr, X = me_X, M = 1, Var.calc = 4,
  estimand = "ATT", sample = T, replace = F
)
summary(me_out)
```

```
##
## Estimate...  -7.4149
## SE.........  1.4634
## T-stat.....  -5.0669
## p.val......  0.00000040443
##
## Original number of observations..............  606
## Original number of treated obs...............  188
## Matched number of observations...............  188
## Matched number of observations  (unweighted).  188
```

```
me_out_md <- rbind(me_df[me_out$index.treated,], me_df[me_out$index.control,])
```

Bonus: Nearest Neighbor Matching with the Matching Package

```r
# Calculate propensity scores
me_glm <- glm(kuse ~ male + black + age97 + pcged97 + mratio96 + pcg_adc,
  family = binomial, data = me_df
)

# Nearest neighbor matching with MatchIt::matchit()
me_MatchIt <- MatchIt::matchit(
  kuse ~ male + black + age97 + pcged97 + mratio96 + pcg_adc,
  method = "nearest",
  distance = "glm",
  estimand = "ATT",
  data = me_df,
  m.order = "data",
  replace = F
)

# Nearest neighbor matching with Matching::Match()
me_Matching <- Matching::Match(
  Y = me_df$pcss97,
  Tr = me_df$kuse,
  X = me_glm$fitted,
  M = 1,
  replace = F,
  estimand = "ATT",
  distance.tolerance = 0,
  ties = F,
  sample = F
)

# Verify that the results are identical
cobalt::bal.tab(me_MatchIt, weights = me_Matching)
```

```
## Balance Measures
##              Type Diff.matchit Diff.Match
## distance Distance       0.9028     0.9028
## male       Binary      -0.0213    -0.0213
## black      Binary       0.1755     0.1755
## age97     Contin.       0.0642     0.0642
## pcged97   Contin.      -0.4191    -0.4191
## mratio96  Contin.      -0.5793    -0.5793
## pcg_adc   Contin.       0.3581     0.3581
##
## Effective sample sizes
##         Control Treated
## All         418     188
## matchit     188     188
## Match       188     188
```

Appendix A: IMBALANCE Stata Module in R

```r
imbalance <- function(df, varname, treatname, blockname) {

  # dx
  df2 <- df %>%
    group_by({{ treatname }}) %>%
    summarise(
      m_x = mean({{ varname }}),
      sd_x = sd({{ varname }}),
      .groups = "drop"
    )
  mxt <- df2[2, 2]
  mxc <- df2[1, 2]
  s2xt <- df2[2, 3]^2
  s2xc <- df2[1, 3]^2
  sx <- sqrt((s2xt + s2xc) / 2)
  dx <- as.numeric(abs(mxt - mxc) / sx)

  # dx
  df3 <- df %>%
    group_by({{ blockname }}, {{ treatname }}) %>%
    summarise(
      m_x = mean({{ varname }}),
      sd_x = sd({{ varname }}),
      n = n(),
      .groups = "drop"
    )

  mxc <- as.numeric(mean(filter(df3, {{ treatname }} == 0)$m_x))
  mxt <- as.numeric(mean(filter(df3, {{ treatname }} == 1)$m_x))
  dxm_num <- abs(mxt - mxc)
  dxm <- as.numeric(dxm_num / sx)

  return(list(dx = dx, dxm = dxm))
}
```

Example Usage

```r
imbalance(df, mratio96, kuse, fm)
```

Appendix B: HODGESL Stata Module in R

```
hodgesl <- function(dataname, varname, blockname, treatname) {
  blockname_str <- deparse(substitute(blockname))
  set.seed(1000)
  renamed_file <- dataname %>%
    filter(!is.na({{ blockname }}))
  r1 <- renamed_file %>%
    group_by({{ blockname }}) %>%
    summarise(m_y = mean({{ varname }}), .groups = "drop") %>%
    arrange({{ blockname }})
  r2 <- renamed_file %>%
    group_by({{ blockname }}, {{ treatname }}) %>%
    summarise(
      mean_y = mean({{ varname }}),
      n = n(), .groups = "drop"
    ) %>%
    mutate(mean_diff = ifelse({{ treatname }} == 1,
      ((n + lag(n)) / sum(n)) * (mean_y - lag(mean_y)), NA
    )) %>%
    mutate(tx_effect = sum(mean_diff, na.rm = T)) %>%
    mutate(i = row_number()) %>%
    slice(1) %>%
    select(tx_effect, i)
  fm_results <- renamed_file %>%
    group_by({{ blockname }}, {{ treatname }}) %>%
    summarise(
      mean_y = mean({{ varname }}),
      n = n(), .groups = "drop"
    )
  r3 <- renamed_file %>%
    group_by({{ blockname }}, {{ treatname }}) %>%
    summarise(m_or_n = n(), .groups = "drop") %>%
    arrange({{ blockname }}, {{ treatname }}) %>%
    mutate(mi = ifelse({{ treatname }} == 0, m_or_n,
      ifelse({{ treatname }} == 1, NA, NA)
    )) %>%
    mutate(ni = ifelse({{ treatname }} == 1, m_or_n,
      ifelse({{ treatname }} == 0, NA, NA)
    )) %>%
    mutate(Ni = ni + lag(mi)) %>%
    mutate(mi = ifelse(is.na(mi), lag(mi), mi)) %>%
    filter(!is.na(Ni)) %>%
    mutate(factor = (mi * ni) / (Ni * (Ni - 1))) %>%
    select({{ blockname }}, factor) %>%
    arrange({{ blockname }})
  r4 <- renamed_file %>%
    arrange({{ blockname }}) %>%
    left_join(r1, by = blockname_str) %>%
    mutate(dy = {{ varname }} - m_y) %>%
    arrange(dy) %>%
    mutate(rk = row_number()) %>%
    arrange({{ blockname }})
  r4a <- r4 %>%
    filter({{ treatname }} != 0) %>%
    group_by({{ blockname }}) %>%
    summarise(wsi = sum(rk), .groups = "drop")
  r5 <- r4 %>%
    group_by({{ blockname }}) %>%
    summarise(ki_ = mean(rk), .groups = "drop")
  r6 <- r4 %>%
    filter({{ treatname }} != 0) %>%
    group_by({{ blockname }}) %>%
    summarise(ni = n(), .groups = "drop") %>%
    arrange({{ blockname }}) %>%
    left_join(r5, by = blockname_str) %>%
    mutate(E_wsi = ni * ki_) %>%
    arrange({{ blockname }})
  r7 <- r4 %>%
    arrange({{ blockname }}) %>%
    left_join(r5, by = blockname_str) %>%
    mutate(k = (rk - ki_)^2) %>%
    group_by({{ blockname }}) %>%
    summarise(ss_kd_i = sum(k), .groups = "drop") %>%
    arrange({{ blockname }})
  results <- r3 %>%
    arrange({{ blockname }}) %>%
    left_join(r7, by = blockname_str) %>%
    left_join(r6, by = blockname_str) %>%
    left_join(r4a, by = blockname_str) %>%
    mutate(
      var_wsi = factor * ss_kd_i,
      var = sum(var_wsi),
      sum_Ewsi = sum(E_wsi),
      ws = sum(wsi),
      HL_mean = ws - sum_Ewsi,
      HL_se = sqrt(var),
      z = HL_mean / HL_se,
      p = 1 - pnorm(abs(z))
    ) %>%
    select(HL_mean, HL_se, z, p) %>%
    slice(1) %>%
    mutate(i = row_number()) %>%
    left_join(r2, by = "i")
  return(results)
}
```

Session Info

```r
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26100)
##
## Matrix products: default
##   LAPACK version 3.12.1
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: Asia/Taipei
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] Matching_4.10-15       MASS_7.3-65            WeightIt_1.5.0
##  [4] lmtest_0.9-40          zoo_1.8-14             marginaleffects_0.30.0
##  [7] rlang_1.1.6            sandwich_3.1-1         broom_1.0.10
## [10] knitr_1.50             optmatch_0.10.8        gbm_2.2.2
## [13] here_1.0.2             psych_2.5.6            survival_3.8-3
## [16] kableExtra_1.4.0       lubridate_1.9.4        forcats_1.0.0
## [19] stringr_1.5.2          dplyr_1.1.4            purrr_1.1.0
## [22] readr_2.1.5            tidyr_1.3.1            tibble_3.3.0
## [25] ggplot2_4.0.0          tidyverse_2.0.0        MatchIt_4.7.2
## [28] cobalt_4.6.1           sjlabelled_1.2.0       haven_2.5.5
## [31] formatR_1.14           tictoc_1.2.1           pacman_0.5.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1   viridisLite_0.4.2  farver_2.1.2       S7_0.2.0
##  [5] fastmap_1.2.0      digest_0.6.37      timechange_0.3.0   lifecycle_1.0.4
##  [9] magrittr_2.0.4     compiler_4.5.1     tools_4.5.1        utf8_1.2.6
## [13] yaml_2.3.10        data.table_1.17.8  labeling_0.4.3     mnormt_2.1.1
## [17] xml2_1.4.0         RColorBrewer_1.1-3 withr_3.0.2        grid_4.5.1
## [21] scales_1.4.0       tinytex_0.57       insight_1.4.2      cli_3.6.5
## [25] rmarkdown_2.29     crayon_1.5.3       generics_0.1.4     rstudioapi_0.17.1
## [29] tzdb_0.5.0         rlemon_0.2.1       splines_4.5.1      parallel_4.5.1
## [33] vctrs_0.6.5        Matrix_1.7-3       hms_1.1.3          systemfonts_1.2.3
## [37] glue_1.8.0         chk_0.10.0         stringi_1.8.7      gtable_0.3.6
## [41] pillar_1.11.1      htmltools_0.5.8.1  R6_2.6.1           textshaping_1.0.3
## [45] rprojroot_2.1.1    evaluate_1.0.5     lattice_0.22-7     backports_1.5.0
## [49] Rcpp_1.1.0         svglite_2.2.1      nlme_3.1-168       checkmate_2.3.3
## [53] xfun_0.53          pkgconfig_2.0.3
## 51.5 sec elapsed
```