Causal Machine Learning Masterclass

Sessions 4 and 5 Lab:
LTMLE for Longitudinal Data with Intercurrent Events

Mark van der Laan & Rachael Phillips
University of California, Berkeley

3 March 2020

Alan Turing Institute in collaboration with the Centre for Statistical Methodology and London School of Hygiene and Tropical Medicine

Statistics for Clinical & Epidemiological Research (SISCER)
Adapted from materials by Professors David Benkeser, Marco Carone & Larry Kessler: Summer Institute in

The `ltmle` package

The `ltmle` package facilitates doubly-robust estimation about average treatment effects of longitudinal interventions. It is available on CRAN and GitHub.

- A Journal of Statistical Software paper is also available.

Learning objectives for today:

1. motivating time-varying interventions;
2. understanding and executing basic calls to `ltmle`;
3. understanding interface between `ltmle` and `SuperLearner`;
4. executing calls to `ltmle` with censoring;
5. executing calls to `ltmle` for longitudinal treatment rules.

Simulating longitudinal data

To illustrate a general longitudinal setting, let's simulate a data structure with three time-varying interventions.

$$
\begin{aligned}
L_{01} &\sim Bernoulli(p = 0.5) \\
L_{02} &\sim Normal(\mu = 0, \mu = 1) \\
A_0 &\sim Bernoulli(p = \text{expit}(0.2 \cdot L_{01} - 0.2 \cdot L_{02})) \\
L_1 &\sim Normal(\mu = -A_0 + L_{01} - L_{02}, 1) \\
A_1 &\sim Bernoulli(p = \text{expit}(0.2 \cdot A_0 - L_1 + L_{01})) \\
L_2 &\sim Normal(\mu = -A_0 \cdot A_1 + 2 \cdot A_1 - L_{01} + L_1, \sigma^2 = 2) \\
A_2 &\sim Bernoulli(p = expit(A_0 - A_1 + 2 \cdot A_0 \cdot A_1 - A_{01} + 0.2 \cdot L_1 \cdot L_{02})) \\
Y &\sim Normal(\mu = L_{01} \cdot L_{02} \cdot L_2 - A_0 - A_1 - A_2 \cdot A_0 \cdot L_2, \sigma^2 = 2)
\end{aligned}
$$

Exercise 1

1. Generate $n = 500$ samples from the above data generating process.
2. Show the first 6 lines and summary statistics of the data using the `head()` and `summary()` functions, respectively.
3. Describe a study/experiment that could be represented by this data generating system.
4. We are interested in estimating the effect of receiving treatment at all three time points versus receiving control at all three time points. Calculate the true value of the estimand.

Exercise 1 Solution

```
# set seed for reproducibility & set sample size of 500
set.seed(212); n <- 500
# baseline variables
L0 <- data.frame(L01 = rnorm(n), L02 = rbinom(n, 1, 0.5))
# first treatment
gA0 <- plogis(0.2 * L0$L01 - 0.2 * L0$L02)
A0 <- rbinom(n = n, size = 1, prob = gA0)
# intermediate variable at time 1
L1 <- rnorm(n = n, mean = -A0 + L0$L01 - L0$L02, sd = 1)
# second treatment decision
gA1 <- plogis(0.2 * A0 - L1 + L0$L01)
A1 <- rbinom(n = n, size = 1, prob = gA1)
# intermediate variable at time 2
L2 <- rnorm(n = n, mean = -A0*A1 + 2*A1 - L0$L01 + L1, sd = 2)
# third treatment decision
gA2 <- plogis(A0 - A1 + 2*A0*A1 - L0$L01 + 0.2 * L1*L0$L02)
A2 <- rbinom(n = n, size = 1, prob = gA2)
# outcome
Y <- rnorm(n = n, mean = L0$L01 * L0$L02 * L2 - A0 - A1 - A2*A0*L2, sd = 2)
# put into a data frame
full_data <- data.frame(L0, A0 = A0, L1 = L1,
                        A1 = A1, L2 = L2, A2 = A2, Y = Y)
```

Exercise 1 Solution

Take a look at the first six rows of data:

```
head(full_data)
```

```
##          L01 L02 A0         L1 A1         L2 A2          Y
## 1 -0.2391731   0  1 -2.0936558  1 -0.4755031  1 -1.5500339
## 2  0.6769356   0  1  1.1531256  0  1.3390966  1 -4.4178134
## 3 -2.4403360   0  0 -1.5562041  1  3.7861115  1 -3.7538463
## 4  1.2408845   0  0 -0.2899494  1  1.2289257  0  0.7606519
## 5 -0.3265144   1  1 -3.7865679  1 -2.5503459  1  1.1983679
## 6  0.1544909   1  1 -1.2827057  1  3.7191556  1 -3.5444271
```

Exercise 1 Solution

```
compute_truth <- function(n = 1e6, a0 = 1, a1 = 1, a2 = 1){
    set.seed(212)
    L0 <- data.frame(L01 = rnorm(n), L02 = rbinom(n, 1, 0.5))
    A0 <- rep(a0, n)
    L1 <- rnorm(n = n, mean = -A0 + L0$L01 - L0$L02, sd = 1)
    A1 <- rep(a1, n)
    L2 <- rnorm(n = n, mean = -A0*A1 + 2*A1 - L0$L01 + L1, sd = 2)
    A2 <- rep(a2, n)
    # outcome
    Y <- rnorm(n = n, mean = L0$L01 * L0$L02 * L2 - A0 - A1 - A2*A0*L2, sd = 2)
    # put into a data frame
    return(mean(Y))
}
```

- True value of $E[Y(1,1,1)] = -1.5$.
- True value of $E[Y(0,0,0)] = 0$.

Simulating longitudinal data with missingness

Often, participants are lost-to-follow-up during the course of the study. Here, we add some right-censoring to our data, and `BinaryToCensoring()` helps to properly format these nodes.

```
set.seed(12)
# censoring prior to time 1 (1 = censored)
gC1 <- plogis(-2 + 0.05 * L0$L01)
C1 <- rbinom(n = n, size = 1, prob = gC1)
# censoring prior to time 2 (1 = censored)
gC2 <- plogis(-3 + 0.05 * A0 + 0.025 * L1 - 0.025 * L0$L02)
C2 <- rbinom(n = n, size = 1, prob = gC2)
# censoring prior to time 3 (1 = censored)
gC3 <- plogis(-3.5 + 0.05*A0*A1 - 0.025*L2 + 0.025 * L1)
C3 <- rbinom(n = n, size = 1, prob = gC3)
# make a cumulative indicator of censoring
anyC1 <- C1 == 1; anyC2 <- C1 == 1 | C2 == 1
anyC3 <- C1 == 1 | C2 == 1 | C3 == 1
# censored data set
cens_data <- data.frame(L0, A0 = A0,
                C1 = ltmle::BinaryToCensoring(is.censored = C1),
                L1 = ifelse(anyC1, NA, L1), A1 = ifelse(anyC1, NA, A1),
                C2 = ltmle::BinaryToCensoring(is.censored = ifelse(anyC1, NA, C2)),
                L2 = ifelse(anyC2, NA, L2), A2 = ifelse(anyC2, NA, A2),
                C3 = ltmle::BinaryToCensoring(is.censored = ifelse(anyC2, NA, C3)),
                Y = ifelse(anyC3, NA, Y))
```

Missing data

```
head(cens_data, 9)
```

```
##          L01 L02 A0         C1          L1 A1         C2
## 1 -0.2391731   0  1 uncensored -2.0936558  1 uncensored
## 2  0.6769356   0  1 uncensored  1.1531256  0 uncensored
## 3 -2.4403360   0  0   censored          NA NA       <NA>
## 4  1.2408845   0  0 uncensored -0.2899494  1 uncensored
## 5 -0.3265144   1  1 uncensored -3.7865679  1 uncensored
## 6  0.1544909   1  1 uncensored -1.2827057  1 uncensored
## 7  1.0368712   1  1 uncensored  0.6649321  1 uncensored
## 8 -0.7796077   1  0 uncensored -1.3935843  1 uncensored
## 9  0.6212641   1  1 uncensored -1.5369416  1 uncensored
##           L2 A2         C3           Y
## 1 -0.4755031  1 uncensored -1.5500339
## 2  1.3390966  1 uncensored -4.4178134
## 3         NA NA       <NA>          NA
## 4  1.2289257  0 uncensored  0.7606519
## 5 -2.5503459  1 uncensored  1.1983679
## 6  3.7191556  1 uncensored -3.5444271
## 7 -1.6484044  1 uncensored -2.9017656
## 8  0.1423414  0 uncensored -3.8194847
## 9 -4.1821615  1 uncensored  1.2824355
```

Basic calls to `ltmle`

A rundown of the most important options for the `ltmle` function:

- `data` = `data.frame` where the order of the columns corresponds to the time-ordering of variables (important!);
- `Anodes` = names of treatment nodes;
- `Cnodes` = names of censoring nodes;
- `Lnodes` = names of time-varying covariate nodes;
- `SL.library` = `list` with named entries `Q` and `g` specifying super learner libraries for the iterated outcome regressions and propensity scores;
- `abar` = binary vector of length `length(Anodes)` or `list` of length 2 to contrast treatments;
- `gbounds` = a vector of lower and upper bounds on estimated propensity scores;
- `stratify` = if `TRUE` then regressions are performed separately for each `abar`. If `FALSE` (default), then regressions are pooled over `abar`.

For survival analysis:

- `Ynodes` = names or indexes of time-varying outcome nodes;
- `survivalOutcome` = `TRUE` if outcome is event that occurs only once, `FALSE` otherwise.
- Alternatively, see package `survtmle`.

For treatment rules:

- `rule` function that can be applied to each row of data, which should return a numeric vector of treatment assignments of length `length(Anodes)`.

Basic calls to ltmle

```
library(ltmle)
library(SuperLearner)
```

Let's start by making a simple call to ltmle and parsing the output.

- Get counterfactual mean for all treatment and all control.
- The super learner library for propensity scores and outcome regressions uses polynomial multivariate adaptive regression splines, logistic regression, and intercept-only regression.
- We fit regressions pooled over all treatments.

```
set.seed(123)
ltmle_fit1 <- ltmle(
    data = full_data,
    Anodes = c("A0", "A1", "A2"),
    Lnodes = c("L01","L02","L1","L2"),
    Ynodes = "Y",
    SL.library = list(Q = c("SL.earth", "SL.glm", "SL.mean"),
                      g = c("SL.earth", "SL.glm", "SL.mean")),
    stratify = FALSE, abar = list(treatment = c(1,1,1),
                                  control = c(0,0,0))
    )
```

Basic calls to `ltmle`

```
## Some Ynodes are not in [0, 1], and Yrange was NULL, so all Y nodes are
## being transformed to (Y-min.of.all.Ys)/range.of.all.Ys
```

- Feature/flaw of `ltmle`: outcomes automatically scaled to be between 0 and 1.
- In general, this is fine. It prevents regression estimators from extrapolating outside the range of the observed data.
- However, super learner is called with `family = binomial()`, even though the outcome assumes values continuously between 0 and 1. This may cause issues with some wrappers (e.g., `SL.glmnet`).

```
## Qform not specified, using defaults:
## formula for L1:
## Q.kplus1 ~ L01 + L02 + A0
## formula for L2:
## Q.kplus1 ~ L01 + L02 + A0 + L1 + A1
## formula for Y:
## Q.kplus1 ~ L01 + L02 + A0 + L1 + A1 + L2 + A2
```

- `Qform` indicates what variables to include in each outcome regression. If `NULL` (default) it includes all variables from previous time points.
- Confusingly, not an indication that a `glm` was used for the outcome regressions.
- See the function documentation for more.

Basic calls to `ltmle`

```
## gform not specified, using defaults:
## formula for A0:
## A0 ~ L01 + L02
## formula for A1:
## A1 ~ L01 + L02 + A0 + L1
## formula for A2:
## A2 ~ L01 + L02 + A0 + L1 + A1 + L2
```

- `gform` indicates what variables to include in each propensity score. If `NULL` (default) it includes all variables from previous time points.
- Confusingly, not an indication that a `glm` was used for the propensity scores.
- See the function documentation for more.

```
## Warning messages:
## In predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(type == :
## prediction from a rank-deficient fit may be misleading
```

- Current version of `ltmle` is doing something silly to cause this error – safe to ignore.
- A fix is pending.

Basic calls to ltmle

The summary method provides results.

```
summary(ltmle_fit1)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = full_data, Anodes = c("A0", "A1", "A2"), Lnodes = c("L01",
##     "L02", "L1", "L2"), Ynodes = "Y", abar = list(treatment = c(1,
##     1, 1), control = c(0, 0, 0)), stratify = FALSE, SL.library = list(Q = c("SL.
##     "SL.glm", "SL.mean"), g = c("SL.earth", "SL.glm", "SL.mean")))
##
## Treatment Estimate:
##    Parameter Estimate:  -1.6376
##     Estimated Std Err:  0.24672
##               p-value:  <2e-16
##     95% Conf Interval: (-2.1212, -1.1541)
##
## Control Estimate:
##    Parameter Estimate:  0.16413
##     Estimated Std Err:  0.28925
##               p-value:  <2e-16
##     95% Conf Interval: (-0.40279, 0.73104)
```

Basic calls to `ltmle`

```
##
## Additive Treatment Effect:
##    Parameter Estimate:  -1.8018
##     Estimated Std Err:  0.37996
##               p-value:  2.1168e-06
##     95% Conf Interval: (-2.5465, -1.057)
```

- `Treatment Estimate` pertains to $E[Y(1,1,1)]$.
- `Control Estimate` pertains to $E[Y(0,0,0)]$.
- `Additive Treatment Effect` pertains to $E[Y(1,1,1)] - E[Y(0,0,0)]$.
- All `p-value`'s are of null hypothesis that quantity equals 0.

Unfortunately, the full super learner objects for each regression cannot be accessed from `ltmle_fit1`. However, the weights given to each regression at each time are saved.

Basic calls to ltmle

```
# weights for outcome regressions, because we set stratify = FALSE, the output in
# ltmle_fit1$fit$Q[[1]] is the same as in ltmle_fit1$fit$Q[[2]]
ltmle_fit1$fit$Q[[1]]
```

```
## $L1
##                      Risk       Coef
## SL.earth_All 0.001848613 0.58151911
## SL.glm_All   0.001874259 0.40812164
## SL.mean_All  0.002683676 0.01035926
##
## $L2
##                      Risk       Coef
## SL.earth_All 0.007511131 0.60433728
## SL.glm_All   0.007830406 0.37698630
## SL.mean_All  0.010002381 0.01867641
##
## $Y
##                     Risk      Coef
## SL.earth_All 0.01096039 0.8517972
## SL.glm_All   0.01588644 0.1482028
## SL.mean_All  0.01793702 0.0000000
```

Basic calls to ltmle

```
# weights for propensity scores, because we set stratify = FALSE, the output in
# ltmle_fit1$fit$g[[1]] is the same as in ltmle_fit1$fit$g[[2]]

ltmle_fit1$fit$g[[1]]


## $A0
##                    Risk       Coef
## SL.earth_All 0.2480814 0.00000000
## SL.glm_All   0.2451089 0.92935948
## SL.mean_All  0.2505096 0.07064052
##
## $A1
##                    Risk       Coef
## SL.earth_All 0.1696186 0.21983761
## SL.glm_All   0.1664212 0.73773235
## SL.mean_All  0.2040158 0.04243004
##
## $A2
##                    Risk        Coef
## SL.earth_All 0.1722207 0.440177871
## SL.glm_All   0.1699507 0.552778712
## SL.mean_All  0.2451067 0.007043417
```

Example write-up of LTMLE analysis

Methods

We estimated the average counterfactual outcome if patients received treatment at all three time points versus if patients received control at all three time points using super learning and longitudinal targeted minimum loss-based estimation (van der Laan and Gruber, 2010). This requires estimation of an iterated outcome regression and the probability for treatment at each time point. At each time point, these regressions adjusted for measured patient characteristics prior to that timepoint. At baseline, these characteristics included [. . .]; at the second time point these included [. . .]; at the third time point these included [. . .]. Each regression was estimated using super learning. For the outcome regressions, we estimated the linear combination of candidate regression estimators that minimizes ten-fold cross-validated mean squared-error. We included three candidate regression estimators in the super learner: polynomial multivariate regression splines, main terms quasi-logistic regression, and intercept-only regression. The same set of candidate estimators was used for estimating the probability of treatment at each time point. However, in this case we estimated the logistic-linear combination of regression estimators that minimizes ten-fold cross-validated negative log-likelihood loss. We tested the null hypothesis that the average outcomes were the same under treatment versus control using a two-sided, level 0.05 Wald test with influence function-based standard errors estimates. Analyses were performed using the SuperLearner and ltmle R packages (Polley et al, 2018; Lendle et al 2017).

Example write-up of LTMLE analysis

```
tmp <- summary(ltmle_fit1)
EY1 <- tmp$effect.measures$treatment$estimate
EY1_ci <- tmp$effect.measures$treatment$CI
EY0 <- tmp$effect.measures$control$estimate
EY0_ci <- tmp$effect.measures$control$CI
```

Results

Depending on the number of time points, it may be overwhelming to describe the super learners fit for each regression. It may suffice to provide general statements.

Overall, the super learners for the iterated outcome regressions tended to give the most weight to polynomial multivariate adaptive regression splines, while for the treatment probability the main-terms logistic regression tended to have the most weight (Table 1, Appendix A).

The estimated average counterfactual outcome if patients received treatment at all three time points was -1.64 (95% CI: -2.12, -1.15). On the other hand the estimated average counterfactual outcome if patients received control at all three time points was 0.16 (95% CI: -0.40, 0.73). Our test of the null hypothesis that these two quantities are equal rejected the null hypothesis (p-value $<$ 0.001).

Sensitivity analyses examining super learner performance are more difficult to conduct in these settings, particularly for the iterated outcome regressions.

Example write-up of LTMLE analysis

```
w1 <- formatC(ltmle_fit1$fit$Q[[1]][[1]][,2], digits = 2, format = "f")
w2 <- formatC(ltmle_fit1$fit$Q[[1]][[2]][,2], digits = 2, format = "f")
w3 <- formatC(ltmle_fit1$fit$Q[[1]][[3]][,2], digits = 2, format = "f")
```

Appendix

Iterated outcome regressions and super learner weights

Function name	Description	Time 1	Time 2	Time 3
`SL.glm_All`	main-terms linear regression using all previous variables	0.41	0.38	0.15
`SL.mean_All`	intercept-only regression	0.01	0.02	0.00
`SL.earth_All`	polynomial multivariate adaptive regression splines using all previous variables and "default" tuning parameters	0.58	0.60	0.85

Exercise 2

1. Make a call to `ltmle` using the censored data set.
2. Get counterfactual mean for all treatment and all control.
3. Specify `c("SL.earth", "SL.glm", "SL.mean")` as the super learner library for propensity scores (which now includes censoring) and outcome regressions, which uses polynomial multivariate adaptive regression splines, logistic regression, and intercept-only regression.

Exercise 2 Solution

The specific formatting of `Cnodes` is important. We previously used the helper function `BinaryToCensoring` to properly format these variables.

This procedure fits regressions pooled over all treatments using uncensored observations.

```
set.seed(123)
ltmle_fit2 <- ltmle(
    data = cens_data,
    Anodes = c("A0", "A1", "A2"),
    Lnodes = c("L01","L02","L1","L2"),
    Cnodes = c("C1","C2","C3"),
    Ynodes = "Y",
    SL.library = list(Q = c("SL.earth", "SL.glm", "SL.mean"),
                      g = c("SL.earth", "SL.glm", "SL.mean")),
    stratify = FALSE, abar = list(treatment = c(1,1,1),
                                  control = c(0,0,0))
    )
```

Exercise 2 Solution

```
summary(ltmle_fit2)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = cens_data, Anodes = c("A0", "A1", "A2"), Cnodes = c("C1",
##     "C2", "C3"), Lnodes = c("L01", "L02", "L1", "L2"), Ynodes = "Y",
##     abar = list(treatment = c(1, 1, 1), control = c(0, 0, 0)),
##     stratify = FALSE, SL.library = list(Q = c("SL.earth", "SL.glm",
##         "SL.mean"), g = c("SL.earth", "SL.glm", "SL.mean")))
##
## Treatment Estimate:
##    Parameter Estimate:  -1.6158
##     Estimated Std Err:  0.26973
##               p-value:  <2e-16
##     95% Conf Interval: (-2.1445, -1.0871)
##
## Control Estimate:
##    Parameter Estimate:  0.24192
##     Estimated Std Err:  0.36437
##               p-value:  <2e-16
##     95% Conf Interval: (-0.47224, 0.95607)
```

Exercise 2 Solution

```
##
## Additive Treatment Effect:
##    Parameter Estimate:  -1.8577
##     Estimated Std Err:  0.45346
##               p-value:  4.1895e-05
##     95% Conf Interval: (-2.7465, -0.96896)


## earth glm Y: did not converge after 25 iterations

## glm.fit:  algorithm did not converge
```

- For some regressions, there are few observations with the outcome.
- E.g., `C3 ~ L01 + L02 + A0 + L1 + A1 + L2 + A2` has only 9 censored observations.
- By default, `ltmle` tries use `V = 10` fold cross-validation, which leads to instability.
- Corrections for this are in the works.

Other notes:

- `ltmle_fit2$fit$g` additionally contains super learner risks/weights for censoring.

Dynamic treatment regimes

Suppose we are interested in comparing two treatment regimes:

- Give all patients control until $L_k > -1$, then give treatment.
- E.g., monitor patients until back pain worsens, then give treatment.
- Give all patients control at every time point.

In `ltmle` this is achieved by the `rule` and `regime` options.

- A `rule` is a `function` that looks at a patient's data and outputs a vector of binary treatment assignments for that patient.
- The `regimes` option will is a list of `rules`.

Dynamic treatment regimes

Here we define a `rule` for "give all patients control until $L_k > -1$, then give treatment."

```
rule1 <- function(pt_data){
    # all patients start on control
    A0 <- 0
    # patients get treatment at time 1 if L1 > -1
    # set patients with missing L1 to NA
    if(!is.na(pt_data$L1)){
        A1 <- ifelse(pt_data$L1 > -1, 1, 0)
    }else{
        A1 <- NA
    }
    # patients get treatment at time 2 if L2 > -1
    # set patients with missing L2 to NA
    if(!is.na(pt_data$L1)){
        A2 <- ifelse(pt_data$L2 > -1, 1, 0)
    }else{
        A2 <- NA
    }
    return(c(A0,A1,A2))
}
```

Exercise 3

1. Now try to define a `rule` for give all patients control at every time point.

Exercise 3 Solution

```
rule2 <- function(pt_data){
    # all patients start on control
    A0 <- 0
    # and stay on control unless censored
    A1 <- ifelse(is.na(pt_data$L1), NA, 0)
    A2 <- ifelse(is.na(pt_data$L2), NA, 0)
    return(c(A0,A1,A2))
}
```

Dynamic treatment regimes

We now make a call to `ltmle` using the censored data set.

- Get counterfactual mean for the two treatment rules
- Same super learner and other options as before.

```
set.seed(123)
ltmle_fit3 <- ltmle(
    data = cens_data,
    Anodes = c("A0", "A1", "A2"),
    Lnodes = c("L01","L02","L1","L2"),
    Cnodes = c("C1","C2","C3"),
    Ynodes = "Y", stratify = FALSE,
    SL.library = list(Q = c("SL.earth", "SL.glm", "SL.mean"),
                      g = c("SL.earth", "SL.glm", "SL.mean")),
    rule = list(treatment = rule1, control = rule2)
    )
```

Dynamic treatment regimes

```
summary(ltmle_fit3)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = cens_data, Anodes = c("A0", "A1", "A2"), Cnodes = c("C1",
##     "C2", "C3"), Lnodes = c("L01", "L02", "L1", "L2"), Ynodes = "Y",
##     rule = list(treatment = rule1, control = rule2), stratify = FALSE,
##     SL.library = list(Q = c("SL.earth", "SL.glm", "SL.mean"),
##         g = c("SL.earth", "SL.glm", "SL.mean")))
##
## Treatment Estimate:
##    Parameter Estimate:  -0.22419
##     Estimated Std Err:  0.31623
##               p-value:  <2e-16
##     95% Conf Interval: (-0.84399, 0.3956)
##
## Control Estimate:
##    Parameter Estimate:  0.38042
##     Estimated Std Err:  0.36485
##               p-value:  <2e-16
##     95% Conf Interval: (-0.33469, 1.0955)
```

Dynamic treatment regimes

```
##
## Additive Treatment Effect:
##    Parameter Estimate:  -0.60461
##     Estimated Std Err:  0.45254
##               p-value:  0.18154
##     95% Conf Interval: (-1.4916, 0.28235)
```

- The output under `Treatment` is whatever `rule` was first in the list.
- The output under `Control` is whatever `rule` was second in the list.