Modern Statistical Learning Methods for Observational Biomedical Data

Lab 3: TMLE for multiple timepoint interventions

David Benkeser
Emory Univ.

Marco Carone
Univ. of Washington

Larry Kessler
Univ. of Washington

Module 14
5th Annual Summer Institute for Statistics in Clinical Research

07/22/2018

Installing R packages

To follow along with this demonstration, you will need several R packages.

- Packages are freely available from the Comprehensive R Archive Network (CRAN).
- Packages are downloaded to your local computer via the `install.packages` function.
- Packages are loaded into your current R session via `require` or `library`

```r
# these packages are needed to execute the demo
pkgs <- c("ltmle","SuperLearner")
# see what packages are currently installed
installed_pacakges <- row.names(installed.packages())
# loop over the needed packages
for(p in pkgs){
  # check if package is installed
  already_installed <- p %in% installed_pacakges
  # if not already installed, install it
  if(!already_installed){
    install.packages(p)
  }
  # and load package
  library(p, character.only = TRUE)
}
```

Simulating data

We will use the data discussed in Chapter 5 to illustrate key ideas.

$$\begin{aligned}
Y \mid A_1 = a_1, L_1 = \ell_1, A_0 = a_0 &\sim \text{Normal}(1 + a_1 + 2\ell_1, 1) \\
A_1 \mid L_1 = \ell_1, A_0 = a_0 &\sim \text{Bernoulli}(\text{expit}(-1 + \ell_1 + a_0)) \\
L_1 \mid A_0 = a_0 &\sim \text{Normal}(1 + a_0, 1) \\
A_0 &\sim \text{Bernoulli}(0.5)
\end{aligned}$$

```r
# set a seed for reproducibility
set.seed(212)
n <- 5000
A0 <- rbinom(n, size = 1, p = 0.5)
L1 <- rnorm(n, mean = A0 + 1, sd = 1)
A1 <- rbinom(n, size = 1, p = plogis(-1 + L1 + A0))
Y <- rnorm(n, mean = 1 + A1 + 2 * L1, 1)
```

Failure of naive approach to causal inference

In this example, we demonstrate numerically the failure of the standard, regression-based approach to causal inference.

```r
# fit a regression of Y ~ A1 + L1 + A0
fit <- glm(Y ~ A1 + L1 + A0)
# show results
fit
```

```
##
## Call:  glm(formula = Y ~ A1 + L1 + A0)
##
## Coefficients:
## (Intercept)           A1           L1           A0
##     1.04814      0.98394      1.98862     -0.04264
##
## Degrees of Freedom: 4999 Total (i.e. Null);  4996 Residual
## Null Deviance:       34720
## Residual Deviance: 4767  AIC: 13960
```

The fitted regression gives an estimate of the conditional mean of Y,

$$\widehat{E}[Y \mid A_1, L_1, A_0] = 1.05 + 0.98A_1 + 1.99L_1 + -0.04A_0 \ .$$

Failure of the naive approach to causal inference

Can causal effects be read off the regression of Y on (A_1, L_1, A_0)?

<u>Effect of differing A_1 values but same A_0 value:</u>

$$
\begin{aligned}
&E[Y(1,1) - Y(1,0)] \;=\; 1 \\
&\widehat{E}[Y \mid A_1 = 1, L_1, A_0 = 1] - \widehat{E}[Y \mid A_1 = 0, L_1, A_0 = 1] \\
&\quad = 1.05 + 0.98 \times 1 + 1.99L_1 + -0.04 \times 1 - (1.05 + 0.98 \times 0 + 1.99L_1 + -0.04 \times 1) \\
&\quad = 0.98
\end{aligned}
$$

$$
\begin{aligned}
&E[Y(0,1) - Y(0,0)] \;=\; 1 \\
&\widehat{E}[Y \mid A_1 = 1, L_1, A_0 = 0] - \widehat{E}[Y \mid A_1 = 0, L_1, A_0 = 0] \\
&\quad = 1.05 + 0.98 \times 1 + 1.99L_1 + -0.04 \times 0 - (1.05 + 0.98 \times 0 + 1.99L_1 + -0.04 \times 0) \\
&\quad = 0.98
\end{aligned}
$$

Failure of the naive approach to causal inference

Can causal effects be read off the regression of Y on (A_1, L_1, A_0)?

Effect of differing A_1 values but same A_0 value:

$$
\begin{aligned}
&E[Y(1,1) - Y(0,1)] \;=\; 2\\
&\widehat{E}[Y \mid A_1 = 1, L_1, A_0 = 1] - \widehat{E}[Y \mid A_1 = 0, L_1, A_0 = 0]\\
&\quad = 1.05 + 0.98 \times 1 + 1.99L_1 + -0.04 \times 1 - (1.05 + 0.98 \times 1 + 1.99L_1 + -0.04 \times 0)\\
&\quad = -0.04
\end{aligned}
$$

$$
\begin{aligned}
&E[Y(1,0) - Y(0,0)] \;=\; 2\\
&\widehat{E}[Y \mid A_1 = 0, L_1, A_0 = 1] - \widehat{E}[Y \mid A_1 = 0, L_1, A_0 = 0]\\
&\quad = 1.05 + 0.98 \times 0 + 1.99L_1 + -0.04 \times 1 - (1.05 + 0.98 \times 0 + 1.99L_1 + -0.04 \times 0)\\
&\quad = -0.04
\end{aligned}
$$

Failure of naive approach to causal inference

Can causal effects be read off the regression of Y on (A_1, L_1, A_0)?

Effect of differing A_1 and A_0 values:

$$
\begin{aligned}
E[Y(1,1) - Y(0,0)] &= 3 \\
\widehat{E}[Y \mid A_1 = 1, L_1, A_0 = 1] - \widehat{E}[Y \mid A_1 = 0, L_1, A_0 = 0]& \\
= 1.05 + 0.98 \times 1 + 1.99L_1 + -0.04 \times 1 - (1.05 + 0.98 \times 0 + 1.99L_1 + -0.04 \times 0)& \\
= 0.98 + -0.04 = 0.94&
\end{aligned}
$$

$$
\begin{aligned}
E[Y(1,0) - Y(0,1)] &= 1 \\
\widehat{E}[Y \mid A_1 = 0, L_1, A_0 = 1] - \widehat{E}[Y \mid A_1 = 1, L_1, A_0 = 0]& \\
= 1.05 + 0.98 \times 0 + 1.99L_1 + -0.04 \times 1 - (1.05 + 0.98 \times 1 + 1.99L_1 + -0.04 \times 0)& \\
= -0.04 - 0.98 = -1.02&
\end{aligned}
$$

Illustration of G-computation

We will now demonstrate that the G-computation formula gives correct answers.

Goal: compute $E[E[Y \mid A_1 = a_1, L_1, A_0 = a_0] \mid A_0 = a_0]$ for different values of (a_0, a_1).

A helpful way to think about regression quantities

$$E[\underbrace{Z}_{\text{outcome}} \mid \underbrace{S = s}_{\text{stratification}}, \underbrace{C}_{\text{covariates}}]$$

Considering the inner expectation, we have

$$E[\underbrace{Y}_{\text{outcome}} \mid \underbrace{A_1 = a_1, A_0 = a_0}_{\text{stratification}}, \underbrace{L_1}_{\text{covariate}}] .$$

Illustration of G-computation

For example, if $a_0 = 1, a_1 = 1$,

```
# full data.frame
full_data <- data.frame(A0 = A0, L1 = L1, A1 = A1, Y = Y)
# subset data to observations with A0 = 1 & A1 = 1
data_11 <- subset(full_data, A0 == 1 & A1 == 1)
# fit regression of Y ~ L1
fit_11 <- glm(Y ~ L1, data = data_11)
fit_11
```

```
##
## Call:  glm(formula = Y ~ L1, data = data_11)
##
## Coefficients:
## (Intercept)           L1
##       1.899        2.031
##
## Degrees of Freedom: 2140 Total (i.e. Null);  2139 Residual
## Null Deviance:       9740
## Residual Deviance: 1959  AIC: 5892
```

Illustration of G-computation

The fitted regression gives us the estimate

$$\widehat{E}[Y \mid A_1 = 1, L_1, A_0 = 1] = 1.9 + 2.03L_1 \ .$$

Now, we need to estimate the outer expectation,

$$E[\underbrace{E[\ Y \mid A_1 = 1, L_1, A_0 = 1]}_{\text{outcome}} \mid \ \underbrace{A_0 = 1}_{\text{stratification}}\ ]$$

I.e., regression with outcome $1.9 + 2.03L_1$ in observations with $A_0 = 0$ and no covariates.

Illustration of G-computation

```
# get predicted value for everyone
full_data$Q2n_11 <- predict(fit_11, newdata = full_data)
# subset data to observations with A0 = 1
data_1 <- subset(full_data, A0 == 1)
# fit regression
fit_1 <- glm(Q2n_11 ~ 1, data = data_1)
# intercept is estimate of E[Y(1,1)]
fit_1
```

```
##
## Call:  glm(formula = Q2n_11 ~ 1, data = data_1)
##
## Coefficients:
## (Intercept)
##       5.963
##
## Degrees of Freedom: 2532 Total (i.e. Null);  2532 Residual
## Null Deviance:       10060
## Residual Deviance: 10060     AIC: 10690
```

Exercise

Use G-computation to obtain estimates of $E[Y(0,1)], E[Y(1,0)]$, and $E[Y(0,0)]$

```
# subset data to observations with A0 = a0 & A1 = a1

# fit regression of Y ~ L1 in A0/A1 subset data

# get predicted value for everyone

# subset data to observations with A0 = a0

# fit intercept-only regression in A0 subset data

# intercept is estimate of E[Y(a0,a1)]
```

Solution

Here is a function that computes the answer for any given ‘a0‘, ‘a1‘

```r
cfmean_gcomp <- function(a0, a1, full_data){
    # subset data to observations with A0 = a0 & A1 = a1
    data_a0a1 <- subset(full_data, A0 == a0 & A1 == a1)
    # fit regression of Y ~ L1 in A0/A1 subset data
    fit_a0a1 <- glm(Y ~ L1, data = data_a0a1)
    # get predicted value for everyone
    full_data$Q2n_a0a1 <- predict(fit_a0a1, newdata = full_data)
    # subset data to observations with A0 = a0
    data_a0 <- subset(full_data, A0 == a0)
    # fit intercept-only regression in A0 subset data
    fit_a0 <- glm(Q2n_a0a1 ~ 1, data = data_a0)
    # intercept is estimate of E[Y(a0,a1)]
    return(as.numeric(fit_a0$coefficients))
}
# evaluate the function
EY11_gcomp <- cfmean_gcomp(a0 = 1, a1 = 1, full_data)
EY10_gcomp <- cfmean_gcomp(a0 = 1, a1 = 0, full_data)
EY01_gcomp <- cfmean_gcomp(a0 = 0, a1 = 1, full_data)
EY00_gcomp <- cfmean_gcomp(a0 = 0, a1 = 0, full_data)
```

Solution

Here are the estimated counterfactual means.

```
# should be ~ 6, 5, 4, 3
round(c(EY11_gcomp, EY10_gcomp, EY01_gcomp, EY00_gcomp), 2)
```

```
## [1] 5.96 4.96 4.04 3.03
```

Illustration of IPTW

Similarly, the IPTW identification result can be used. In this example, we can write

$$E[Y(a_0, a_1)] = E\left[\frac{I(A_0 = a_0)I(A_1 = a_1)}{P(A_0 = a_0)P(A_1 = a_1 \mid A_0 = a_0, L_1)} \; Y \right] .$$

This result suggests using the estimate

$$\widehat{E}[Y(a_0, a_1)] = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{I(A_{0i} = a_0)I(A_{1i} = a_1)}{\widehat{P}(A_0 = a_0)\widehat{P}(A_1 = a_1 \mid A_0 = a_0, L_1 = L_{1i})} \; Y_i \right] .$$

A helpful way to think about regression quantities

$$P[\; \underbrace{Z = z}_{\substack{\text{binary outcome} \\ I(Z=z)}} \;\mid\; \underbrace{S = s}_{\text{stratification}} \;,\; \underbrace{\;C\;}_{\text{covariates}} \;]$$

Illustration of IPTW

Here is a function that computes the IPTW estimator for any given a0, a1.

```r
cfmean_iptw <- function(a0, a1, full_data){
    # subset data to observations with A0 = a0
    data_a0 <- subset(full_data, A0 == a0)
    # fit logistic regression of I(A1 = a1) ~ L1 in a0 subset
    ps_a1 <- glm(I(A1 == a1) ~ L1, data = data_a0, family = binomial())
    # get predicted value for everybody
    full_data$phat_a1 <- predict(ps_a1, newdata = full_data,
                                 type = 'response')
    # fit regression of I(A0 = a0) ~ 1 in full_data
    ps_a0 <- glm(I(A0 == a0) ~ 1, data = full_data, family = binomial())
    # get predicted value for everybody
    full_data$phat_a0 <- predict(ps_a0, newdata = full_data,
                                 type = 'response')
    # compute iptw estimator
    EYa0a1 <- with(full_data, mean(
      as.numeric(A0 == a0) * as.numeric(A1 == a1) / (phat_a0 * phat_a1) * Y
    ))
    # intercept is estimate of E[Y(a0,a1)]
    return(EYa0a1)
}
```

Illustration of IPTW

```
# evaluate the function
EY11_iptw <- cfmean_iptw(a0 = 1, a1 = 1, full_data)
EY10_iptw <- cfmean_iptw(a0 = 1, a1 = 0, full_data)
EY01_iptw <- cfmean_iptw(a0 = 0, a1 = 1, full_data)
EY00_iptw <- cfmean_iptw(a0 = 0, a1 = 0, full_data)
# should be ~ 6,5,4,3
round(c(EY11_iptw, EY10_iptw, EY01_iptw, EY00_iptw),2)
```

```
## [1] 5.96 5.28 4.02 2.99
```

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. understanding and executing basic calls to ‘ltmle‘;
2. understanding interface between ‘ltmle‘ and ‘SuperLearner‘;
3. executing calls to ‘ltmle‘ with censoring;
4. executing calls to ‘ltmle‘ for longitudinal treatment rules.

Simulated data

To illustrate a more general setting, we simulate a data structure with three treatments.

```r
# 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)
```

Simulated data

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
```

We are interested in estimating the effect of receiving treatment at all three time points versus receiving control at all three time points.

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

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

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.ea
##     "SL.glm", "SL.mean"), g = c("SL.earth", "SL.glm", "SL.mean")))
##
## Treatment Estimate:
##    Parameter Estimate:  -1.6317
##     Estimated Std Err:  0.2474
##               p-value:  <2e-16
##     95% Conf Interval: (-2.1166, -1.1468)
##
## Control Estimate:
##    Parameter Estimate:  0.17039
##     Estimated Std Err:  0.28555
##               p-value:  <2e-16
##     95% Conf Interval: (-0.38928, 0.73006)
```

Basic calls to `ltmle`

```
##
## Additive Treatment Effect:
##    Parameter Estimate:  -1.8021
##     Estimated Std Err:  0.37748
##               p-value:  1.8059e-06
##     95% Conf Interval: (-2.5419, -1.0622)
```

- `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 class 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 0.001685760 0.399302939
## SL.glm_All   0.001663461 0.593744229
## SL.mean_All  0.002460856 0.006952832
##
## $L2
##                      Risk        Coef
## SL.earth_All 0.007431811 0.526947556
## SL.glm_All   0.007509260 0.464259051
## SL.mean_All  0.009449858 0.008793393
##
## $Y
##                     Risk       Coef
## SL.earth_All 0.01089596 0.85480228
## SL.glm_All   0.01598388 0.05266457
## SL.mean_All  0.01793251 0.09253314
```

Basic class 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 0.2503784 0.0000000
## SL.glm_All   0.2463051 0.8354287
## SL.mean_All  0.2504346 0.1645713
##
## $A1
##                    Risk       Coef
## SL.earth_All 0.1705573 0.07001174
## SL.glm_All   0.1661950 0.88354936
## SL.mean_All  0.2041659 0.04643890
##
## $A2
##                    Risk        Coef
## SL.earth_All 0.1701692 0.491707245
## SL.glm_All   0.1697660 0.501847091
## SL.mean_All  0.2453319 0.006445664
```

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

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 -1.63 (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 0.17 (95% CI: -0.39, 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

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	0.59	0.46	0.05
`SL.mean_All`	intercept-only regression	0.01	0.01	0.09
`SL.earth_All`	polynomial multivariate adaptive regression splines using all previous variables and "default" tuning parameters	0.40	0.53	0.85

Missing data

Often, participants are lost-to-follow-up during the course of the study. Here, we add some right-censoring to our data.

```r
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 = BinaryToCensoring(is.censored = C1),
               L1 = ifelse(anyC1, NA, L1), A1 = ifelse(anyC1, NA, A1),
               C2 = BinaryToCensoring(is.censored = ifelse(anyC1, NA, C2)),
               L2 = ifelse(anyC2, NA, L2), A2 = ifelse(anyC2, NA, A2),
               C3 = BinaryToCensoring(is.censored = ifelse(anyC2, NA, C3)),
               Y = ifelse(anyC3, NA, Y))
```

Missing data

```r
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
```

Missing data

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

- Get counterfactual mean for all treatment and all control.
- The super learner library for propensity scores (which now includes censoring!) and outcome regressions uses polynomial multivariate adaptive regression splines, logistic regression, and intercept-only regression.
- The specific formatting of `Cnodes` is important. The helper function `BinaryToCensoring` can help properly format these variables.
- We fit 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))
    )
```

Missing data

```r
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:  -1.6413
##     Estimated Std Err:  0.2687
##               p-value:  <2e-16
##     95% Conf Interval: (-2.168, -1.1147)
##
## Control Estimate:
##    Parameter Estimate:  0.22323
##     Estimated Std Err:  0.34766
##               p-value:  <2e-16
##     95% Conf Interval: (-0.45817, 0.90462)
##
```

Missing data

```
## Additive Treatment Effect:
##    Parameter Estimate:  -1.8645
##     Estimated Std Err:  0.43971
##               p-value:  2.2314e-05
##     95% Conf Interval: (-2.7264, -1.0027)
```

```
## 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))
}
```

Dynamic treatment regimes

Now, we define a `rule` for give all patients control at every time point.

```r
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

```r
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:  -0.25533
##     Estimated Std Err:  0.29283
##               p-value:  <2e-16
##     95% Conf Interval: (-0.82926, 0.31861)
##
## Control Estimate:
##    Parameter Estimate:  0.19432
##     Estimated Std Err:  0.34299
##               p-value:  <2e-16
##     95% Conf Interval: (-0.47793, 0.86656)
##
```

Dynamic treatment regimes

```
## Additive Treatment Effect:
##    Parameter Estimate:  -0.44964
##     Estimated Std Err:  0.41686
##               p-value:  0.28075
##     95% Conf Interval: (-1.2667, 0.3674)
```

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