Project 8 Template

Brenda Sciepura

May 09, 2024, 23:24

```
# Add to this package list for additional SL algorithms
pacman::p_load(
  tidyverse,
  ggthemes,
  ltmle,
  tmle,
  SuperLearner,
  tidymodels,
  caret,
  dagitty,
  ggdag,
  here)

heart_disease <- read_csv('/Users/brenda/github/Computational-Social-Science-Projects/Project 8/heart_di
```

```
## Rows: 10000 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): age, sex_at_birth, simplified_race, college_educ, income_thousands...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Introduction

Heart disease is the leading cause of death in the United States, and treating it properly is an important public health goal. However, it is a complex disease with several different risk factors and potential treatments. Physicians typically recommend changes in diet, increased exercise, and/or medication to treat symptoms, but it is difficult to determine how effective any one of these factors is in treating the disease. In this project, you will explore SuperLearner, Targeted Maximum Likelihood Estimation (TMLE), and Longitudinal Targeted Maximum Likelihood Estimation (LTMLE). Using a simulated dataset, you will explore whether taking blood pressure medication reduces mortality risk.

Data

This dataset was simulated using R (so it does not come from a previous study or other data source). It contains several variables:

- **blood_pressure_medication**: Treatment indicator for whether the individual took blood pressure medication (0 for control, 1 for treatment)
- **mortality**: Outcome indicator for whether the individual passed away from complications of heart disease (0 for no, 1 for yes)
- **age**: Age at time 1
- **sex_at_birth**: Sex assigned at birth (0 female, 1 male)
- **simplified_race**: Simplified racial category. (1: White/Caucasian, 2: Black/African American, 3: Latinx, 4: Asian American,
  5: Mixed Race/Other)
- **income_thousands**: Household income in thousands of dollars
- **college_educ**: Indicator for college education (0 for no, 1 for yes)
- **bmi**: Body mass index (BMI)
- **chol**: Cholesterol level
- **blood_pressure**: Systolic blood pressure
- **bmi_2**: BMI measured at time 2
- **chol_2**: Cholesterol measured at time 2
- **blood_pressure_2**: BP measured at time 2
- **blood_pressure_medication_2**: Whether the person took treatment at time period 2

For the "SuperLearner" and "TMLE" portions, you can ignore any variable that ends in "_2", we will reintroduce these for LTMLE.

SuperLearner

Modeling

Fit a SuperLearner model to estimate the probability of someone dying from complications of heart disease, conditional on treatment and the relevant covariates. Do the following:

1. Choose a library of at least 5 machine learning algorithms to evaluate. **Note**: We did not cover how to hyperparameter tune constituent algorithms within SuperLearner in lab, but you are free to do so if you like (though not required to for this exercise).
2. Split your data into train and test sets.
3. Train SuperLearner
4. Report the risk and coefficient associated with each model, and the performance of the discrete winner and SuperLearner ensemble
5. Create a confusion matrix and report your overall accuracy, recall, and precision

```r
# Fit SuperLearner Model

## sl lib

listWrappers()
```

```
## All prediction algorithm wrappers in SuperLearner:

##  [1] "SL.bartMachine"      "SL.bayesglm"         "SL.biglasso"
##  [4] "SL.caret"            "SL.caret.rpart"      "SL.cforest"
##  [7] "SL.earth"            "SL.gam"              "SL.gbm"
## [10] "SL.glm"              "SL.glm.interaction"  "SL.glmnet"
## [13] "SL.ipredbagg"        "SL.kernelKnn"        "SL.knn"
## [16] "SL.ksvm"             "SL.lda"              "SL.leekasso"
## [19] "SL.lm"               "SL.loess"            "SL.logreg"
## [22] "SL.mean"             "SL.nnet"             "SL.nnls"
## [25] "SL.polymars"         "SL.qda"              "SL.randomForest"
## [28] "SL.ranger"           "SL.ridge"            "SL.rpart"
## [31] "SL.rpartPrune"       "SL.speedglm"         "SL.speedlm"
## [34] "SL.step"             "SL.step.forward"     "SL.step.interaction"
## [37] "SL.stepAIC"          "SL.svm"              "SL.template"
## [40] "SL.xgboost"

##
## All screening algorithm wrappers in SuperLearner:

## [1] "All"
## [1] "screen.corP"           "screen.corRank"        "screen.glmnet"
## [4] "screen.randomForest"   "screen.SIS"            "screen.template"
## [7] "screen.ttest"          "write.screen.template"
```

```r
## Train/Test split

# initial split
# ----------
heart_split <-
  initial_split(heart_disease, prop = 3/4) # create initial split (tidymodels)

# Training
# ----------
train <-
  # Declare the training set with rsample::training()
  training(heart_split)

# y_train
y_train <-
  train %>%
  pull(mortality)

# x_train
x_train <-
  train %>%
```

```
  # drop the target variable
  select(-mortality)

# Testing
# ----------
test <-
  # Declare the training set with rsample::training()
  testing(heart_split)

# y test
y_test <-
  test %>%
  pull(mortality)

# x test
x_test <-
  test %>%
  select(-mortality)

## Train SuperLearner

# set seed
set.seed(12345)

# multiple models
# ----------
sl = SuperLearner(Y = y_train,
                  X = x_train,
                  family = binomial(),
                  SL.library = c('SL.mean',
                                 'SL.glmnet',
                                 'SL.glm'))

## Risk and Coefficient of each model
sl$cvRisk
```

```
##   SL.mean_All SL.glmnet_All    SL.glm_All
##     0.2496975     0.2372463     0.2375087
```

```
sl$coef
```

```
##   SL.mean_All SL.glmnet_All    SL.glm_All
##     0.0000000     0.8847275     0.1152725
```

```
## Discrete winner and superlearner ensemble performance

sl$cvRisk[which.min(sl$cvRisk)]
```

```
## SL.glmnet_All
##     0.2372463
```

```r
## Confusion Matrix

# predictions
# ----------
preds <-
  predict(sl,             # use the superlearner not individual models
          x_test,         # prediction on test set
          onlySL = TRUE)  # use only models that were found to be useful (had weights)


# start with y_test
validation <-
  y_test %>%
  # add our predictions - first column of predictions
  bind_cols(preds$pred[,1]) %>%
  # rename columns
  rename(obs = `...1`,       # actual observations
         pred = `...2`) %>% # predicted prob
  # change pred column so that obs above .5 are 1, otherwise 0
  mutate(pred = ifelse(pred >= .5,
                           1,
                           0))
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
```

```r
# confusion matrix
# ----------
cm <- caret::confusionMatrix(as.factor(validation$pred),
                       as.factor(validation$obs))

# Print accuracy
print(paste("Accuracy:", cm$overall["Accuracy"]))
```

```
## [1] "Accuracy: 0.6024"
```

```r
# Print recall (Sensitivity)
print(paste("Recall (Sensitivity):", cm$byClass["Recall"]))
```

```
## [1] "Recall (Sensitivity): 0.293202293202293"
```

```r
# Print precision
print(paste("Precision:", cm$byClass["Precision"]))
```

```
## [1] "Precision: 0.732106339468303"
```

```r
# Print F1-score
print(paste("F1-score:", cm$byClass["F1"]))
```

```
## [1] "F1-score: 0.418713450292398"
```

Discussion Questions

1. Why should we, in general, prefer the SuperLearner ensemble to the discrete winner in cross-validation? Or in other words, what is the advantage of "blending" algorithms together and giving them each weights, rather than just using the single best algorithm (with best being defined as minimizing risk)?

- By blending multiple base learners in a SuperLearner ensamble, we achieve better generalization compared to a single model. This is especially useful when we're analyzing noisy datasets where we find that no single model performs well on its own.
- The SuperLearner ensemble aggregates predictions from multiple models, which can help mitigate overfitting, which is more likely when we rely on an algorithm identified through cross-validation.
- In a complex data scenario, it's usually difficult to know which algorithm will perform best. With SuperLearner, we are able to better deal with model misspecification, because if a single base learner performs poorly because of misspecification or overfitting, the overall selection of models in the ensemble can still have good predictions.
- Because we observe heterogeneity in datasets and different algorithms have different strengths to deal with different aspects of that heterogeneity, by using a multitude of algorithms in the SuperLearner ensemble, we can leverage the strengths of each model to better adapt to the underlying structure of the data.

Targeted Maximum Likelihood Estimation

Causal Diagram

TMLE requires estimating two models:

1. The outcome model, or the relationship between the outcome and the treatment/predictors, $P(Y|(A,W)$.
2. The propensity score model, or the relationship between assignment to treatment and predictors $P(A|W)$

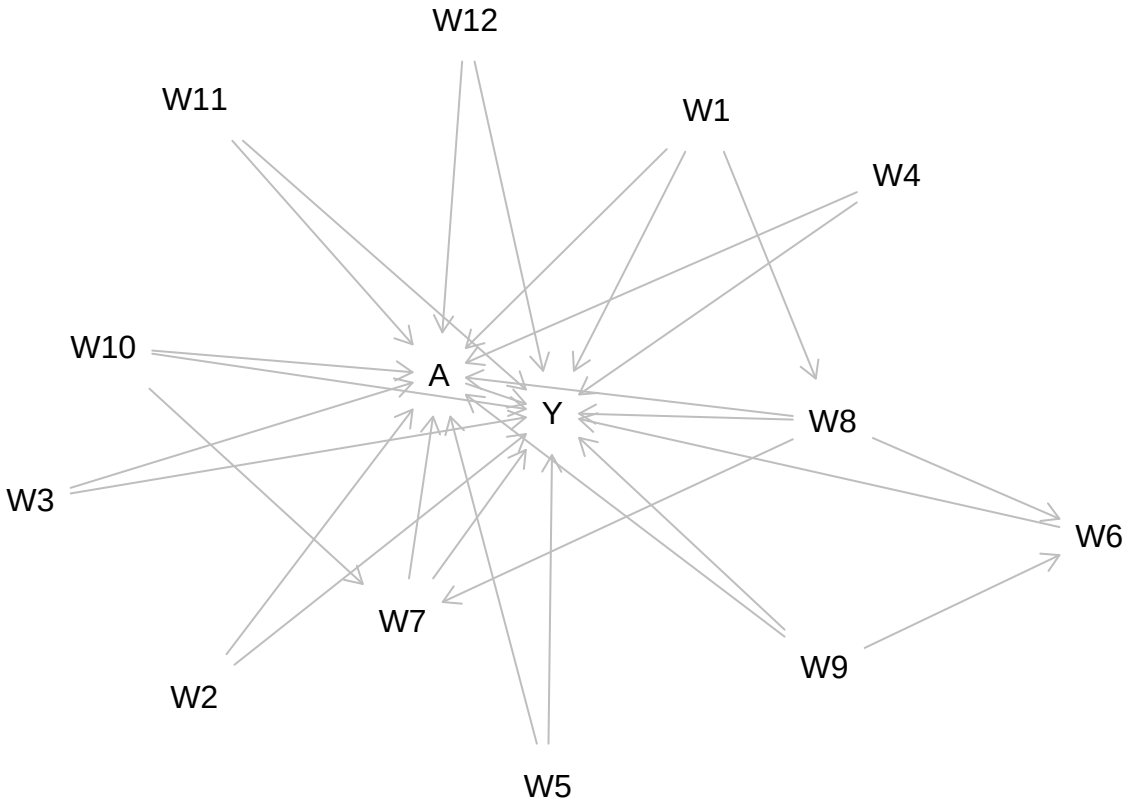
Using ggdag and daggity, draw a directed acylcic graph (DAG) that describes the relationships between the outcome, treatment, and covariates/predictors. Note, if you think there are covariates that are not related to other variables in the dataset, note this by either including them as freestanding nodes or by omitting them and noting omissions in your discussion.

```
# DAG for TMLE

# Create a DAG using dagify
# ----------
dag <- dagify(Y ~ A + W1 + W2 + W3 + W4 + W5 + W6 + W7 + W8 + W9 + W10 + W11 + W12,
              A ~ W1 + W2 + W3 + W4 + W5 + W7 + W8 + W9 + W10 + W11 + W12,
              W6 ~ W9, ## BMI time 1 influences BMI time 2
              W7 ~ W10, ## Chol time 1 influences Chol time 2
              W7 ~ W8, ## Chol affects blood pressure
              W6 ~ W8, ## BMI affects blood pressure
              W8 ~ W1, ## Blood pressure time 1 influences blood pressure time 2
              A ~ W12, ## Blood pressure medication time 1 to blood pressure medication time 2
              exposure = "A",
              outcome = "Y")

plot(dag)
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set your
```



Reference:

- Y: mortality.
- A: blood pressure medication time 1.
- W1: age time 1 (affects treatment because we can argue that the older you're the more likely you are to forget to take the medication, and it affects the outcome because mortality risk increases with age).
- W2: sex (we could argue that females are more likely to be compliant with taking medication, and we can argue that males have a higher association with mortality).
- W3: race (some races have a higher mortality risk, I don't really know about likelihood of taking medication).
- W4: income (positively associated with taking medication, negatively associated with mortality).
- W5: college education (positively associated with taking medication, negatively associated with mortality).
- W6: BMI time 1 (perhaps not associated with treatment (?), positively associated with mortality).
- W7: Cholesterol time 1 (could go either way with treatment, it can be that you're used to taking medication and you're more likely to take this medication or it could go the other way, positively associated with mortality).
- W8: Blood pressure time 1.
- W9: BMI time 2.
- W10: Cholesterol time 2.
- W11: Blood pressure time 2.
- W12: Blood pressure medication time 2.

This DAG could represent a scenario with a potential causal identification, if we met these assumptions:

- No unmeasured confounding influencing both the exposure and outcome. We're assuming no unobserved confounders.

- No Feedback Loops: No variables in the model are influenced by the outcome. We don't see that any variables here are influenced by the outcome.

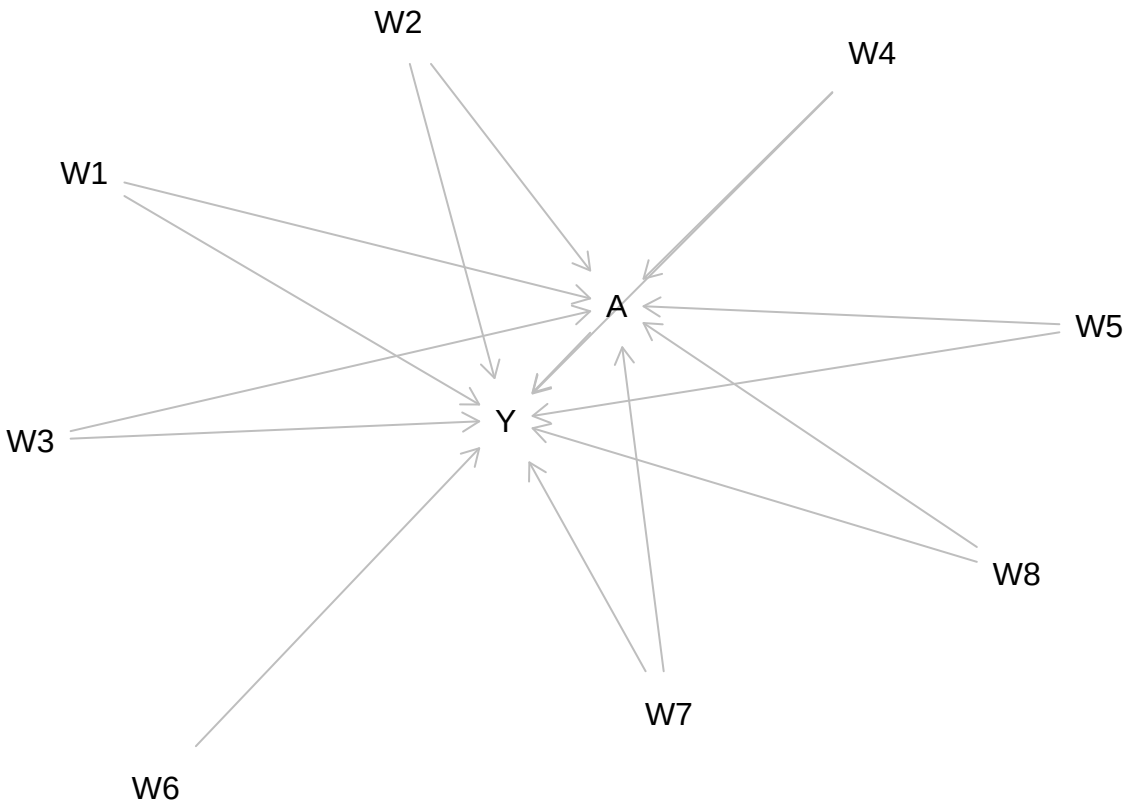
- No Collider Bias: There are no variables influenced by both the exposure and another variable in the model that could introduce bias if conditioned upon. We do have this problem. For instance, blood pressure time 2 is influenced by blood pressure medication and blood presure taking at time 1. We end up doing a simplified model. Also blood pressure medication at time 2 is influenced also by blood presure at time 1 and blood pressure medication at time 2. We're doing a simplified model in which we're removing time 2 variables

```
# DAG for TMLE

# Create a DAG using dagify
# ----------
dag <- dagify(Y ~ A + W1 + W2 + W3 + W4 + W5 + W6 + W7 + W8,
              A ~ W1 + W2 + W3 + W4 + W5 + W7 + W8,
              exposure = "A",
              outcome = "Y")

plot(dag)
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set your
```



TMLE Estimation

Use the `tmle` package to estimate a model for the effect of blood pressure medication on the probability of mortality. Do the following:

1. Use the same SuperLearner library you defined earlier
2. Use the same outcome model and propensity score model that you specified in the DAG above. If in your DAG you concluded that it is not possible to make a causal inference from this dataset, specify a simpler model and note your assumptions for this step.
3. Report the average treatment effect and any other relevant statistics

```
# specify which SuperLearner algorithms we want to use
# ----------
sl_libs <- c('SL.glmnet', 'SL.glm')

# Prepare data for SuperLearner/TMLE
# ----------
# Mutate Y, A for outcome and treatment, use tax, age, and crim as covariates
data_obs <-
  heart_disease %>%
  rename(A = blood_pressure_medication, Y = mortality) %>%
  select(Y, A, sex_at_birth, simplified_race, bmi, chol, blood_pressure, college_educ, income_thousands

# Outcome
# ----------
Y <-
  data_obs %>%
  pull(Y)

# Covariates
# ----------
W_A <-
  data_obs %>%
  select(-Y)

# Fit SL for Q step, initial estimate of the outcome
# ----------
Q <- SuperLearner(Y = Y,                # outcome
                  X = W_A,              # covariates + treatment
                  family = binomial(),  # binominal bc outcome is binary
                  SL.library = sl_libs) # ML algorithms

## Now that we have trained our model, we need to create predictions for three different scenarios:

## 1. Predictions assuming every unit had its observed treatment status.
## 2. Predictions assuming every unit was treated.
## 3. Predictions assuming every unit was control.

# observed treatment
# ----------
Q_A <- as.vector(predict(Q)$pred)
```

```r
# if every unit was treated (pretending every unit was treated)
# ----------
W_A1 <- W_A %>% mutate(A = 1)
Q_1 <- as.vector(predict(Q, newdata = W_A1)$pred)

# if everyone was control (pretending every unit was not treated)
# ----------
W_A0 <- W_A %>% mutate(A = 0)
Q_0 <- as.vector(predict(Q, newdata = W_A0)$pred)
```

We can take our predictions and put them all into one dataframe:

```r
# combine all predictions into one dataframe
# ----------
dat_tmle <- tibble(Y = Y,
                   A = W_A$A,
                   Q_A,
                   Q_0,
                   Q_1)
# view
head(dat_tmle)
```

```
## # A tibble: 6 x 5
##       Y     A   Q_A   Q_0   Q_1
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1     0 0.588 0.588 0.264
## 2     0     0 0.546 0.546 0.232
## 3     1     0 0.570 0.570 0.250
## 4     0     0 0.588 0.588 0.265
## 5     0     0 0.558 0.558 0.241
## 6     1     0 0.550 0.550 0.235
```

```r
# G-computation
# ----------
ate_gcomp <- mean(dat_tmle$Q_1 - dat_tmle$Q_0)
ate_gcomp
```

```
## [1] -0.3176121
```

Discussion Questions

1. What is a "double robust" estimator? Why does it provide a guarantee of consistency if either the outcome model or propensity score model is correctly specified? Or in other words, why does mispecifying one of the models not break the analysis? **Hint**: When answering this question, think about how your introductory statistics courses emphasized using theory to determine the correct outcome model, and in this course how we explored the benefits of matching.

A **double robust** estimator remains consistent (which means it provides unbiased estimates) under two scenarios: - When the outcome model is correctly specified but the propensity score model is not correctly specified. This happens because the estimator relies on the correct specification of the outcome model to adjust for confounding. - When the propensity score model is correctly specified but the outcome model is

not correctly specified. This happens because the estimator uses the correct specification of the propensity score model to adjust for differences in treatment assignment probabilities between treated and untreated units. Because we matched on the estimated propensity scores, the estimator makes sure that our treated and untreated units can be used for a ckeab comparison. This robustness property is useful especially in the context of observational studies where the data-generating process is usually unknown or complex to think about comprehensively.

LTMLE Estimation

Now imagine that everything you measured up until now was in "time period 1". Some people either choose not to or otherwise lack access to medication in that time period, but do start taking the medication in time period 2. Imagine we measure covariates like BMI, blood pressure, and cholesterol at that time for everyone in the study (indicated by a "_2" after the covariate name).

Causal Diagram

Update your causal diagram to incorporate this new information. **Note**: If your groups divides up sections and someone is working on LTMLE separately from TMLE then just draw a causal diagram even if it does not match the one you specified above.

Hint: Check out slide 27 from Maya's lecture, or slides 15-17 from Dave's second slide deck in week 8 on matching.

Hint: Keep in mind that any of the variables that end in "_2" are likely affected by both the previous covariates and the first treatment when drawing your DAG.

```
# DAG for TMLE

# I did this DAG in the previous exercise. Because I found collider bias, I simplified the model by rem
```

LTMLE Estimation

Use the `ltmle` package for this section. First fit a "naive model" that **does not** control for the time-dependent confounding. Then run a LTMLE model that does control for any time dependent confounding. Follow the same steps as in the TMLE section. Do you see a difference between the two estimates?

```
## Naive Model (no time-dependent confounding) estimate
# process data
# ----------
naive_data <-
  heart_disease %>%
  # need to specify W1, W2, etc
  rename(W1 = sex_at_birth, W2 = age, W3 = simplified_race, W4 = college_educ, W5 = income_thousands, W
  select(W1, W2, W3, W4, W5, W6, W7, W8, A, Y)

# implement ltmle
# ----------
naive_estimate  <- ltmle(naive_data, # dataset
                Anodes = "A",   # vector that shows treatment
                Ynodes = "Y",   # vector that shows outcome
                abar = 1)
```

```
## Qform not specified, using defaults:

## formula for Y:

## Q.kplus1 ~ W1 + W2 + W3 + W4 + W5 + W6 + W7 + W8 + A

##

## gform not specified, using defaults:

## formula for A:

## A ~ W1 + W2 + W3 + W4 + W5 + W6 + W7 + W8

##

## Estimate of time to completion: < 1 minute
```

```
## view
naive_estimate
```

```
## Call:
## ltmle(data = naive_data, Anodes = "A", Ynodes = "Y", abar = 1)
##
## TMLE Estimate:  0.2041733
```

```
## LTMLE estimate
# ----------

## process data
ltmle_time_varying <-
  heart_disease %>%
  rename(W1 = sex_at_birth, W2 = age, W3 = simplified_race, W4 = college_educ, W5 = income_thousands,
         W6 = bmi, W7 = blood_pressure, W8 = chol,
         W9 = bmi_2, W10 = blood_pressure_2, W11 = chol_2,
         A1 = blood_pressure_medication, A2 = blood_pressure_medication_2, Y = mortality) %>%
  select(W1, W2, W3, W4, W5, W6, W7, W8, W9, W10, W11, A1, A2, Y)

## estimate
time_varying_estimate <- ltmle(ltmle_time_varying,
                               Anodes=c("A1", "A2"),  # two treatment variables in time 1 and 2
                               Lnodes=c("W6", "W7", "W8", "W9", "W10", "W11"), # L indicator
                               Ynodes="Y",            # outcome
                               abar=c(1, 1),          # treatment indicator in Anodes vector
                               SL.library = sl_libs)
```

```
## Qform not specified, using defaults:

## formula for Y:
```

```
## Q.kplus1 ~ W1 + W2 + W3 + W4 + W5 + W6 + W7 + W8 + W9 + W10 +     W11 + A1 + A2

##

## gform not specified, using defaults:

## formula for A1:

## A1 ~ W1 + W2 + W3 + W4 + W5 + W6 + W7 + W8 + W9 + W10 + W11

## formula for A2:

## A2 ~ W1 + W2 + W3 + W4 + W5 + W6 + W7 + W8 + W9 + W10 + W11 +     A1

##

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

## Estimate of time to completion: 4 to 18 minutes

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

```
# Print accuracy
time_varying_estimate
```

```
## Call:
## ltmle(data = ltmle_time_varying, Anodes = c("A1", "A2"), Lnodes = c("W6",
##     "W7", "W8", "W9", "W10", "W11"), Ynodes = "Y", abar = c(1,
##     1), SL.library = sl_libs)
##
## TMLE Estimate:  0.2945642
```

Discussion Questions

1. What sorts of time-dependent confounding should we be especially worried about? For instance, would we be concerned about a running variable for age the same way we might be concerned about blood pressure measured at two different times?

Time-dependent confounding is a critical consideration in studies examining the relationship between treatments and outcomes over time, since these confounders can alter the observed treatment-outcome association across different time points.

For instance, age is a well-known risk factor for heart disease, directly influencing an individual's susceptibility to the condition. Age can act as a time-dependent confounder if its impact on heart disease risk changes over time. We know that as individuals age, their risk of heart disease may increase. Then, we need to control for age. However, the importance of age as a time-dependent confounder may vary depending on the duration of the study period. In longitudinal studies spanning a decade or more, the potential for age-related changes in heart disease risk to impact the treatment-outcome relationship is considerable. In contrast, if the study duration is relatively short, such as three years, the relevance of age as a time-dependent confounder diminishes. In such cases, other time-varying factors or shorter-term changes in risk factors may exert more substantial influences on the observed treatment effects.

On the other hand, measuring blood pressure medication at multiple time points is crucial for accurately assessing its effect on the outcome. Blood pressure medication is a time-varying exposure, meaning that individuals may start or stop taking medication or change their dosage over time. By measuring medication usage at multiple time points, we can assess how changes in exposure affect the outcome. Other time-dependent confounding variables, such as changes in disease severity, lifestyle factors, or other medications the patient is also taking, may influence both the treatment and the outcome. Measuring the treatment at multiple times allows us to account for these time-varying confounders and obtain more accurate estimates of the true treatment effect.