Project 8: Longitudinal Data

Saron Goitom

05/10/2024

```r
# 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/sarongoitom/git/Computational-Social-Science-Training-Program/Projects
```

```
## 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

```
# Fit SuperLearner Model

## sl lib

## Train/Test split

    # initial_split function from tidymodels/rsample
        hd_split <- initial_split(heart_disease, prop = 3/4)
    # Declare the training set with rsample::training()
        train <- training(hd_split)
    # y_train is medv where medv > 22 is a 1, 0 otherwise
        y_train <- train %>%
          pull(mortality)
    # x_train is everything but the outcome
        x_train <- train %>%
        select(-mortality)

    # Do the same procedure with the test set
        test <- testing(hd_split)
        y_test <- test %>%
          pull(mortality)
        x_test <- test %>%
          select(-mortality)
```

```
## Train SuperLearner and Obtain Risk and Coefficient of Each Model
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"
```

```
#install.packages("ranger")
library(ranger)
sl = SuperLearner(Y = y_train,
                  X = x_train,
                  family = binomial(),
                  SL.library = c('SL.mean',
                                 'SL.glmnet',
                                 'SL.ranger'))
sl
```

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = c("SL.mean",
##     "SL.glmnet", "SL.ranger"))
##
##
##                     Risk       Coef
## SL.mean_All   0.2497778 0.01041054
## SL.glmnet_All 0.2356970 0.41608137
## SL.ranger_All 0.2331563 0.57350809
```

```
## Discrete winner and superlearner ensemble performance
    preds <- predict(sl,
                     x_test,
                     onlySL = TRUE)
    # start with y_test
        validation <- y_test %>%
    # add our predictions
        bind_cols(preds$pred[,1]) %>%
    # rename columns
        rename(obs = `...1`,
               pred = `...2`) %>%
          mutate(pred = ifelse(pred >= .5,
                               1,
                               0))
```

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

```
        head(validation)
```

```
## # A tibble: 6 x 2
##     obs  pred
##   <dbl> <dbl>
## 1     0     1
## 2     1     1
## 3     0     1
## 4     1     1
## 5     1     0
## 6     0     0
```

```r
## Confusion Matrix
caret::confusionMatrix(as.factor(validation$pred),
as.factor(validation$obs))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  397  230
##          1  806 1067
##
##                Accuracy : 0.5856
##                  95% CI : (0.566, 0.605)
##     No Information Rate : 0.5188
##     P-Value [Acc > NIR] : 1.151e-11
##
##                   Kappa : 0.1554
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3300
##             Specificity : 0.8227
##          Pos Pred Value : 0.6332
##          Neg Pred Value : 0.5697
##              Prevalence : 0.4812
##          Detection Rate : 0.1588
##    Detection Prevalence : 0.2508
##       Balanced Accuracy : 0.5763
##
##        'Positive' Class : 0
##
```

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)?

Using a combination of algorithms together is generally preferred over the single best algorithm because the SuperLearner ensemble assigns a weight depending on how they performed in the cross-validation procedure. This allows for minimal risk and error.

Targeted Maximum Likelihood Estimation

Causal Diagram

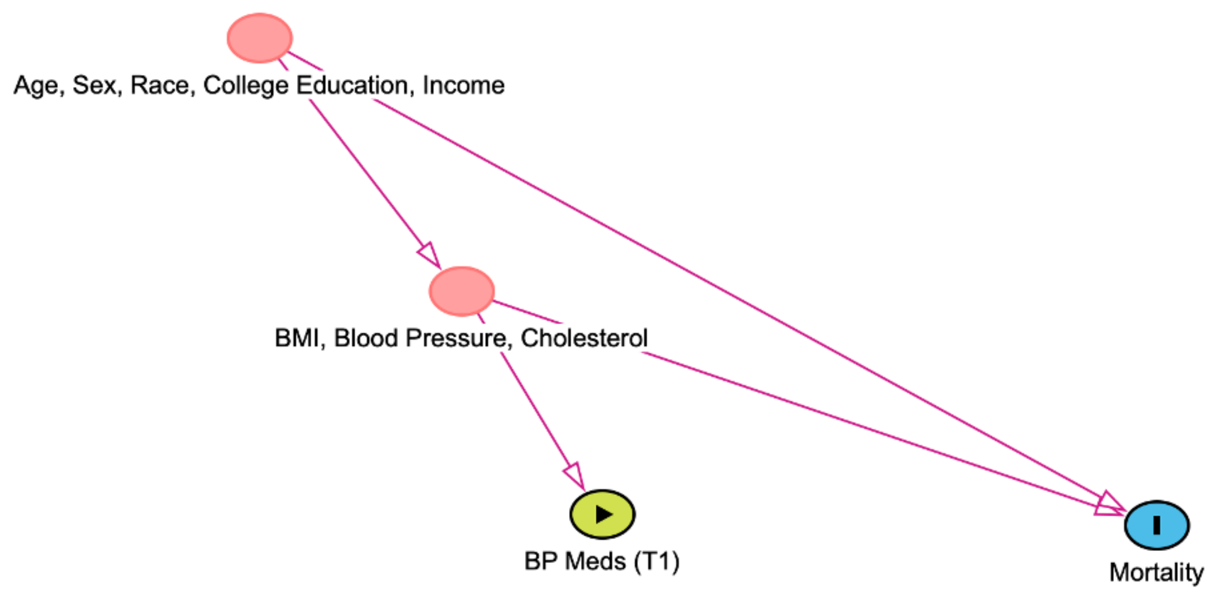
TMLE requires estimating two models:

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.

```r
# DAG for TMLE
knitr::include_graphics("/Users/sarongoitom/Downloads/dag1.png")
```


Age, Sex, Race, College Education, Income
BMI, Blood Pressure, Cholesterol
BP Meds (T1)
Mortality

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

```r
# SuperLearner libraries
sl_libs <- c('SL.glmnet', 'SL.ranger', 'SL.glm')

  # Prepare data for SuperLearner/TMLE
  # Mutate Y, A for outcome and treatment, use bmi, blood pressure, and cholesterol as covariates
      data_obs <- heart_disease %>%
      rename(Y = mortality, A = blood_pressure_medication) %>%
        select(Y, A, bmi, blood_pressure, chol)
  # Data Prep
  # 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,
                        X = W_A,
                        family = binomial(),
                        SL.library = sl_libs)
```

```
# observed treatment
Q_A <- as.vector(predict(Q)$pred)
# if every unit was treated
W_A1 <- W_A %>% mutate(A = 1)
Q_1 <- as.vector(predict(Q, newdata = W_A1)$pred)
# if everyone was control
W_A0 <- W_A %>% mutate(A = 0)
Q_0 <- as.vector(predict(Q, newdata = W_A0)$pred)

dat_tmle <- tibble(Y = Y, A = W_A$A, Q_A, Q_0, Q_1)
head(dat_tmle)
```

```
## # A tibble: 6 x 5
##       Y     A   Q_A   Q_0   Q_1
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1     0 0.730 0.730 0.359
## 2     0     0 0.430 0.430 0.142
## 3     1     0 0.694 0.694 0.149
## 4     0     0 0.469 0.469 0.309
## 5     0     0 0.383 0.383 0.143
## 6     1     0 0.659 0.659 0.141
```

```
## Relevant Statistics

  # ATE part 1 (targeted at the predictions of the outcome)
    ate_gcomp <- mean(dat_tmle$Q_1 - dat_tmle$Q_0)
    ate_gcomp
```

```
## [1] -0.3389553
```

```
  # ATE part 2 (updated with information from propensity score estimates)
    # Probability of Treatment
    # Fluctuation Parameter
    # Update Initial Estimates
    # Compute the Statistical Estimand of Interest
    # Calculate SEs for Cis and p-value

  # Could also get 'targeted' estimate using the TMLE package

A <- W_A$A
```

```
W <- heart_disease %>% select(bmi, chol, blood_pressure)

    tmle_fit <-
      tmle::tmle(Y = Y,
                 A = A,
                 W = W,
                 Q.SL.library = sl_libs,
                 g.SL.library = sl_libs)
    tmle_fit
```

```
##  Marginal mean under treatment (EY1)
##    Parameter Estimate:  0.20725
##    Estimated Variance:  1.4971e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.19966, 0.21483)
##
##  Marginal mean under comparator (EY0)
##    Parameter Estimate:  0.56615
##    Estimated Variance:  2.6275e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.5561, 0.57619)
##
##  Additive Effect
##    Parameter Estimate:  -0.3589
##    Estimated Variance:  4.1181e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.37148, -0.34632)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.31769
##    Estimated Variance:  1.9622e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.32637, -0.30901)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.56016
##    Estimated Variance:  2.7533e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.57044, -0.54987)
```

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 misspecifying 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 means that the final estimate is robust to potential mispecification of either the outcome or propensity score model. Misspecification refers to when there is uncertainty around the correct model form (i.e. appropriate confounders, exposure classification, etc.).

Misspecifying one of the models does not break the analysis because we have two opportunities to control for confounding. In the outcome model, we attempt to build a correctly specified model by turning to theory and the literature around a particular topic to determine appropriate confounders. If this model does not account for all confounding, the propensity score model also accounts for additional confounding that may have not been captured in the outcome model. The propensity score model is the probabilty of treatment given covariates. The idea of using the propensity score as a weight (to either use that to match or include the weight as a covariate in the model) is to create a pseudo-population where there is no confounding.

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.

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?

```r
## Naive Model (no time-dependent confounding) estimate
  data_obs_ltmle <- data_obs %>%
    rename(W1 = bmi, W2 = chol, W3 = blood_pressure) %>%
    select(W1, W2, W3, A, Y)
  result <- ltmle(data_obs_ltmle, Anodes="A", Lnodes=NULL, Ynodes="Y", abar=1, SL.library=sl_libs)
```

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

## formula for Y:

## Q.kplus1 ~ W1 + W2 + W3 + A

##

## gform not specified, using defaults:

## formula for A:
```

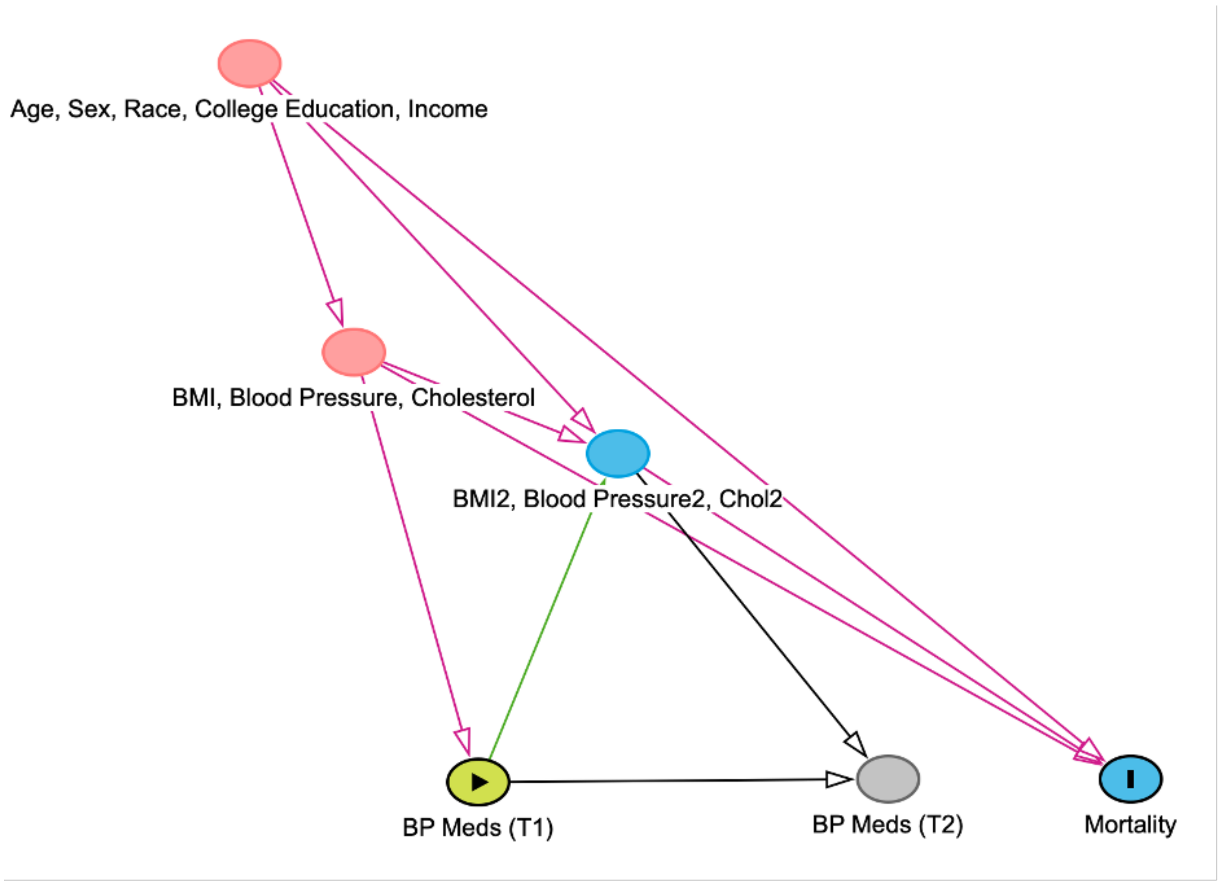

Age, Sex, Race, College Education, Income
BMI, Blood Pressure, Cholesterol
BMI2, Blood Pressure2, Chol2
BP Meds (T1)
BP Meds (T2)
Mortality

Figure 1: A caption

```
## A ~ W1 + W2 + W3

##

## Estimate of time to completion: 2 to 4 minutes
```

```
## LTMLE estimate
  ltmle2 <- heart_disease %>%
    rename(W1 = bmi, W2 = chol, W3 = blood_pressure, L1 = bmi_2, L2 = chol_2, L3 = blood_pressure_2,
                  A1 = blood_pressure_medication, A2 = blood_pressure_medication_2, Y = mortality) %>%
    select(W1, W2, W3, L1, L2, L3, A1, A2, Y)
  result2 <- ltmle(ltmle2, Anodes=c("A1", "A2"), Lnodes=c("L1", "L2", "L3"), Ynodes="Y", abar=c(1, 1), S
```

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

## formula for Y:

## Q.kplus1 ~ W1 + W2 + W3 + L1 + L2 + L3 + A1 + A2

##

## gform not specified, using defaults:

## formula for A1:

## A1 ~ W1 + W2 + W3 + L1 + L2 + L3

## formula for A2:

## A2 ~ W1 + W2 + W3 + L1 + L2 + L3 + A1

##

## Estimate of time to completion: 5 to 23 minutes
```

```
  result
```

```
## Call:
## ltmle(data = data_obs_ltmle, Anodes = "A", Lnodes = NULL, Ynodes = "Y",
##     abar = 1, SL.library = sl_libs)
##
## TMLE Estimate:  0.206392
```

```
  result2
```

```
## Call:
## ltmle(data = ltmle2, Anodes = c("A1", "A2"), Lnodes = c("L1",
##     "L2", "L3"), Ynodes = "Y", abar = c(1, 1), SL.library = sl_libs)
##
## TMLE Estimate:  0.3737909
```

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?

We should be especially worried about time-dependent confounding that have inconsistent changes over time and have a stronger influence on the exposure and outcome. For example, although age is a time-dependent confounder, we know that it increases steadily over time. Conversely, factors like BMI, blood pressure and cholesterol levels may increase/decrease dramatically or not at all. This could have more dramatic and varying effects on both blood pressure medication intake and mortality.