Project 4 SuperLearner and Targeted Maximum Likelihood Estimation

Maggie Ye

```r
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```r
pacman::p_load(
  tidyverse,
  ggthemes,
  ltmle,
  tmle,
  SuperLearner,
  tidymodels,
  caret,
  dagitty,
  ggdag,
  ranger,
  xgboost,
  here)

set.seed(123)

heart_disease <- read_csv('heart_disease_tmle.csv')
```

```
## 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.
```

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
sl_library <- c("SL.mean", "SL.glmnet", "SL.ranger", "SL.xgboost", "SL.glm")

## Train/Test split
heart_split <- initial_split(heart_disease, prop = 3/4)
train <- training(heart_split)
test <- testing(heart_split)

x_train <- select(train, -mortality)
y_train <- train$mortality

x_test <- select(test, -mortality)
y_test <- test$mortality

## Train SuperLearner
sl_model <- SuperLearner(Y = y_train,
                         X = x_train,
                         family = binomial(),
                         SL.library = sl_library)
print(sl_model)
```

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = sl_library)
##
##
##
##                       Risk      Coef
## SL.mean_All    0.2494968 0.0000000
## SL.glmnet_All  0.2354901 0.4045005
## SL.ranger_All  0.2321384 0.5954995
## SL.xgboost_All 0.2509081 0.0000000
## SL.glm_All     0.2360482 0.0000000
```

```r
## Risk and Coefficient of each model
print(sl_model$cvRisk)
```

```
##    SL.mean_All  SL.glmnet_All  SL.ranger_All SL.xgboost_All     SL.glm_All
##      0.2494968      0.2354901      0.2321384      0.2509081      0.2360482
```

```r
print(sl_model$coef)
```

```
##    SL.mean_All  SL.glmnet_All  SL.ranger_All SL.xgboost_All     SL.glm_All
##      0.0000000      0.4045005      0.5954995      0.0000000      0.0000000
```

```r
## Discrete winner and superlearner ensemble performance
discrete_winner <- names(which.min(sl_model$cvRisk))
cat("Discrete winner model:", discrete_winner, "\n")
```

```
## Discrete winner model: SL.ranger_All
```

```
preds <-
predict(sl_model,
x_test,
onlySL = TRUE)

validation <-
  y_test %>%
  bind_cols(preds$pred[,1]) %>%
  rename(obs = `...1`,
         pred = `...2`) %>%
  mutate(pred = ifelse(pred >= .5,
                         1,
                         0))
```

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

```
# view
head(validation)
```

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

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  391  174
##          1  875 1060
##
##                Accuracy : 0.5804
##                  95% CI : (0.5608, 0.5998)
##     No Information Rate : 0.5064
##     P-Value [Acc > NIR] : 6.887e-14
##
##                   Kappa : 0.1666
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3088
##             Specificity : 0.8590
##          Pos Pred Value : 0.6920
##          Neg Pred Value : 0.5478
##              Prevalence : 0.5064
##          Detection Rate : 0.1564
##    Detection Prevalence : 0.2260
##       Balanced Accuracy : 0.5839
##
##        '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)?

Answer: SuperLearner constructs a weighted combination of all the candidate algorithms. By blending multiple algorithms, it tends to average out the noise and errors specific to individual models. This can lead to a reduction in the variance of the predictions, making the ensemble more robust to overfitting than a single model.

Targeted Maximum Likelihood Estimation

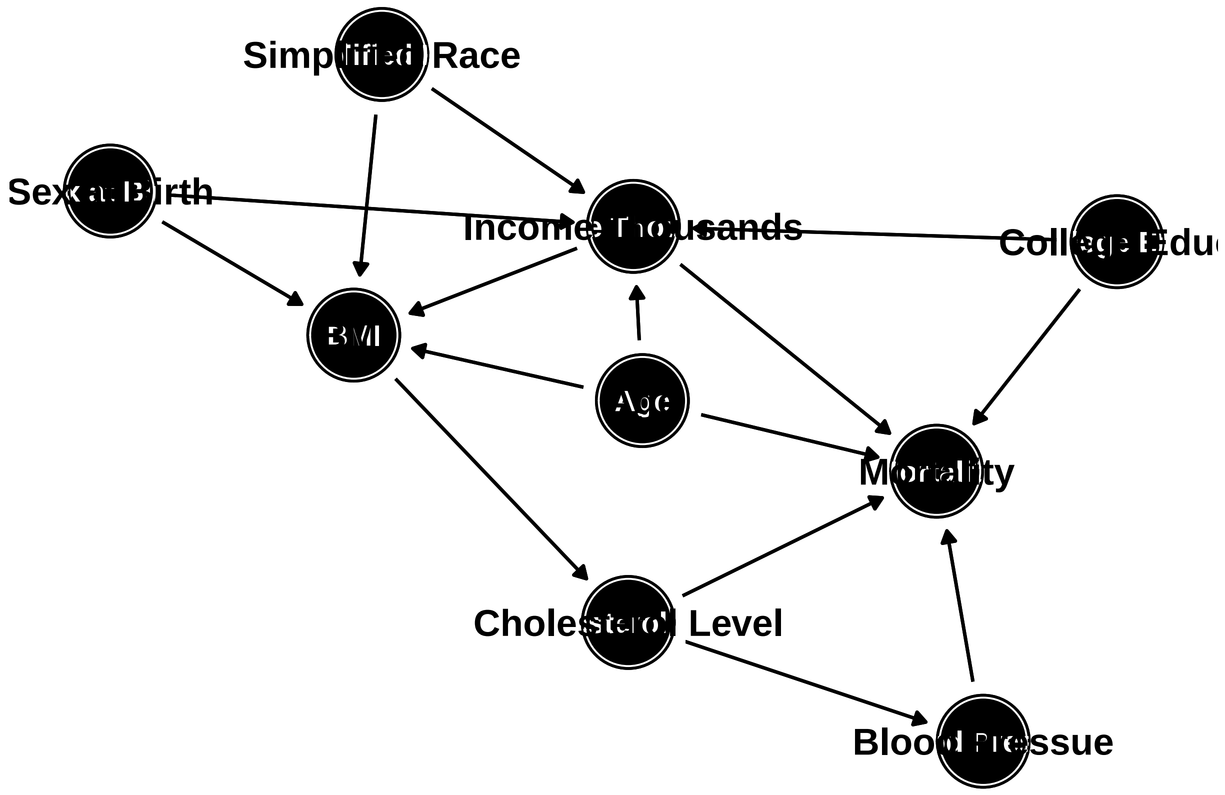
Causal Diagram

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

dag <- dagitty('dag {
  "Sex at Birth" -> "Income Thousands"
  "Sex at Birth" -> "BMI"
  "Age" -> "Income Thousands"
  "Age" -> "BMI"
  "Age" -> "Mortality"
  "Simplified Race" -> "Income Thousands"
  "Simplified Race" -> "BMI"
  "College Educ" -> "Income Thousands"
  "College Educ" -> "Mortality"
  "Income Thousands" -> "BMI"
  "Income Thousands" -> "Mortality"
  "BMI" -> "Cholesterol Level"
  "Cholesterol Level" -> "Mortality"
  "Cholesterol Level" -> "Blood Pressue"
  "Blood Pressue" -> "Mortality"
}')

ggdag(dag, text = TRUE) +
  geom_dag_edges() +
  geom_dag_node(fill = NA, color = "black", shape = 1) +
  geom_dag_text(colour = "black", size = 5) +
  theme_dag_blank()
```



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
tmle_fit <-
  tmle::tmle(Y = heart_disease$mortality,
             A = heart_disease$blood_pressure_medication,
             W = select(heart_disease, -c(income_thousands, college_educ, bmi, chol, blood_pressure)),
             Q.SL.library = sl_library,
             g.SL.library = sl_library)

tmle_fit
```

```
##  Additive Effect
##    Parameter Estimate:  -5.2041e-05
##    Estimated Variance:  1.8715e-10
##               p-value:  0.00014235
##     95% Conf Interval:  (-7.8854e-05, -2.5228e-05)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  0
##    Estimated Variance:  2.0111e-10
##               p-value:  1
##     95% Conf Interval:  (-2.7795e-05, 2.7795e-05)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  0
##    Estimated Variance:  1.8298e-10
##               p-value:  1
##     95% Conf Interval:  (-2.6513e-05, 2.6513e-05)
```

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.

Answer: A "double robust" estimator is one that remains consistent and asymptotically normal if at least one of two critical components—the outcome model or the propensity score model—is correctly specified. The guarantee of consistency, even if only the outcome model or the propensity score model is correctly specified, arises because the estimator compensates for potential errors in one model using the correct specifications of the other. E.g., if the outcome model is accurately specified, it correctly models the conditional expectation of the outcome given the covariates and treatment, ensuring correct estimation despite errors in the propensity score model.

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

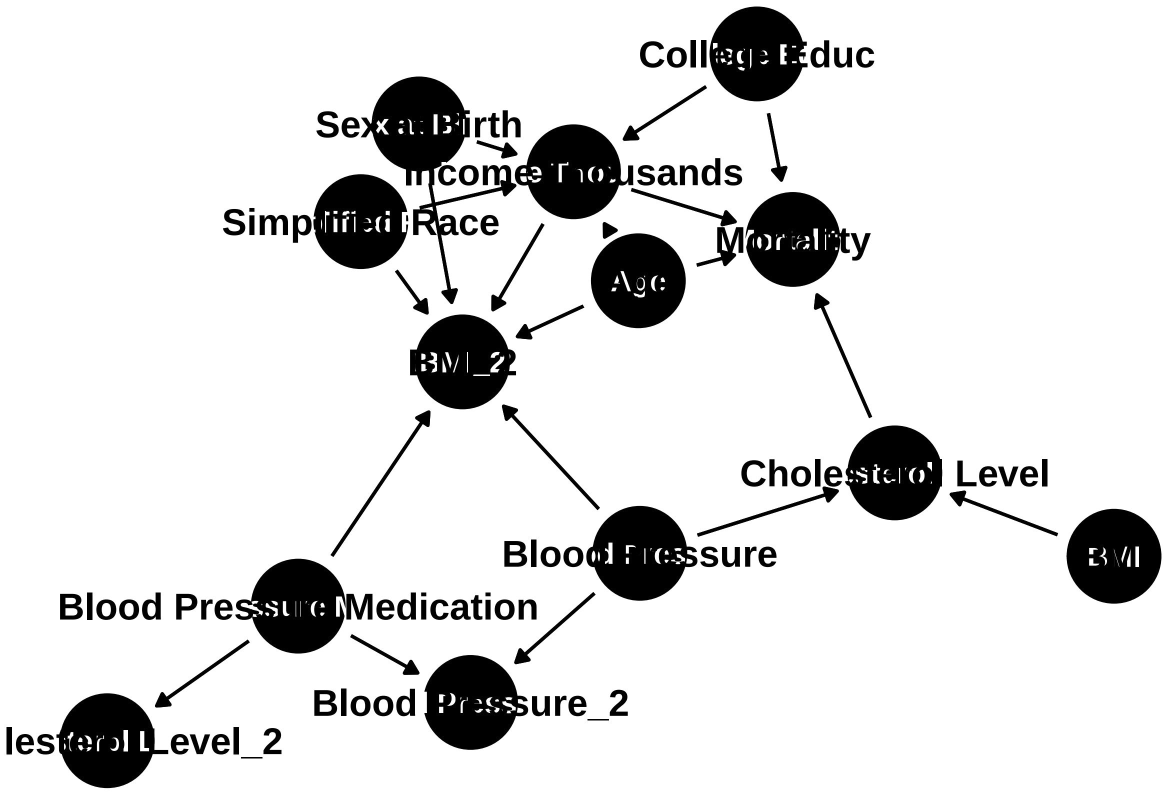
Update your causal diagram to incorporate this new information.

```
# Updated DAG for TMLE

dag2 <- dagitty('dag {
  "Sex at Birth" -> "Income Thousands"
  "Sex at Birth" -> "BMI_2"
  "Age" -> "Income Thousands"
  "Age" -> "BMI_2"
  "Age" -> "Mortality"
  "Simplified Race" -> "Income Thousands"
  "Simplified Race" -> "BMI_2"
  "College Educ" -> "Income Thousands"
  "College Educ" -> "Mortality"
  "Income Thousands" -> "BMI_2"
  "Income Thousands" -> "Mortality"
  "BMI" -> "Cholesterol Level"
  "Cholesterol Level" -> "Mortality"
  "Blood Pressure Medication" -> "BMI_2"
  "Blood Pressure Medication" -> "Cholesterol Level_2"
  "Blood Pressure Medication" -> "Blood Pressure_2"
  "Blood Pressure" -> "Blood Pressure_2"
  "Blood Pressure" -> "Cholesterol Level"
  "Blood Pressure" -> "BMI_2"
}')

ggdag(dag2, text = TRUE) +
  geom_dag_edges() +
  geom_dag_node(fill = "blue", color = NA, size = 10, shape = 10) +
  geom_dag_text(colour = "black", size = 5) +
  theme_dag_blank()
```

```
## Warning: Removed 13 rows containing missing values or values outside the scale range
## (`geom_dag_node()`).
```



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?

```
heart_disease_ltmle <- heart_disease %>%
  rename(W1 = age, W2 = sex_at_birth, W3 = simplified_race) %>%
  select(W1, W2, W3, A = blood_pressure_medication, Y = mortality)

## Naive Model (no time-dependent confounding) estimate

naive_result <- ltmle(heart_disease_ltmle,
                      Anodes = "A",
                      Ynodes = "Y",
                      abar = 1)
```

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

## formula for Y:

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

##

## gform not specified, using defaults:

## formula for A:

## A ~ W1 + W2 + W3

##

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

```
naive_result
```

```
## Call:
## ltmle(data = heart_disease_ltmle, Anodes = "A", Ynodes = "Y",
##     abar = 1)
##
## TMLE Estimate:  0.2485843
```

```
## LTMLE estimate

heart_disease_long <- heart_disease %>%
  rename(W1 = age, W2 = sex_at_birth, W3 = simplified_race,
         A1 = blood_pressure_medication,
         L1 = bmi, L2 = chol, L3 = blood_pressure,
         A2 = blood_pressure_medication_2,
         Y = mortality) %>%
  select(W1, W2, W3, A1, L1, L2, L3, A2, Y)

ltmle_result_long <- ltmle(heart_disease_long,
                           Anodes = c("A1", "A2"),  # two treatment points
                           Lnodes = c("L1", "L2", "L3"),  # time-varying covariates
                           Ynodes = "Y",
                           abar = c(1, 1))
```

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

## formula for L1:

## Q.kplus1 ~ W1 + W2 + W3 + A1

## formula for Y:

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

##

## gform not specified, using defaults:

## formula for A1:

## A1 ~ W1 + W2 + W3

## formula for A2:

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

##
```

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

```
# View results
ltmle_result_long
```

```
## Call:
## ltmle(data = heart_disease_long, Anodes = c("A1", "A2"), Lnodes = c("L1",
##     "L2", "L3"), Ynodes = "Y", abar = c(1, 1))
##
## TMLE Estimate:  0.228895
```

```
# Yes, the estimates are different
```

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 particularly concerned about time-dependent confounding related to variables that change over time and are influenced by past treatment or exposure. For instance, a running variable for age less likely to be influenced by past treatment or exposure in the same way as blood pressure, so it is unlikely to be counfounding. However, blood pressure can be influenced by past treatment or exposure, and changes in blood pressure may affect future treatment decisions and outcomes.Failure to properly account for time-dependent changes in blood pressure can introduce time-dependent confounding.