Project 8

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

pacman::p_load(# Tidyverse packages including dplyr and ggplot2
               e1071,       # svm
               rpart,       # decision trees
               furrr,       # parallel processing
               parallel,    # parallel processing
               ranger,      # fast implementation of random forests
               tidymodels)  # workflow for machine learning

# set seed

heart_disease <- read_csv(here('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.
```

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

```
# 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)


# Fit SuperLearner 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"
```

```r
set.seed(12345)

# LASSO
# ----------
sl_lasso <- SuperLearner(Y = y_train,              # target
                         X = x_train,              # features
                         family = binomial(),      # binomial : 1,0s
                         SL.library = "SL.glmnet") # find the glmnet algo from SL

# view
sl_lasso
```

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = "SL.glmnet")
##
##
##
##                     Risk Coef
## SL.glmnet_All 0.2348619    1
```

```r
# set seed
set.seed(987)

# multiple models
# ----------
sl = SuperLearner(Y = y_train,
                  X = x_train,
                  family = binomial(),
                  # notice these models are concatenated
                  SL.library = c('SL.mean',    # if you just guessed the average - serves as a baseline
                                 'SL.glmnet',
                                 'SL.ranger'))

## Risk and Coefficient of each model
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   0.2497470 0.0000000
## SL.glmnet_All 0.2347693 0.4402096
## SL.ranger_All 0.2327131 0.5597904
```

```
## Discrete winner and superlearner ensemble performance
# Here is the risk of the best model (discrete SuperLearner winner).
# Use which.min boolean to find minimum cvRisk in list
sl$cvRisk[which.min(sl$cvRisk)]
```

```
## SL.ranger_All
##     0.2327131
```

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 the SuperLearner ensemble helps to reduce overfitting. By blending different algorithms, possible overfitting from each one gets averaged out. By that same idea, it is also more robust to model selection issues and complexities in the data.

Targeted Maximum Likelihood Estimation

Causal Diagram

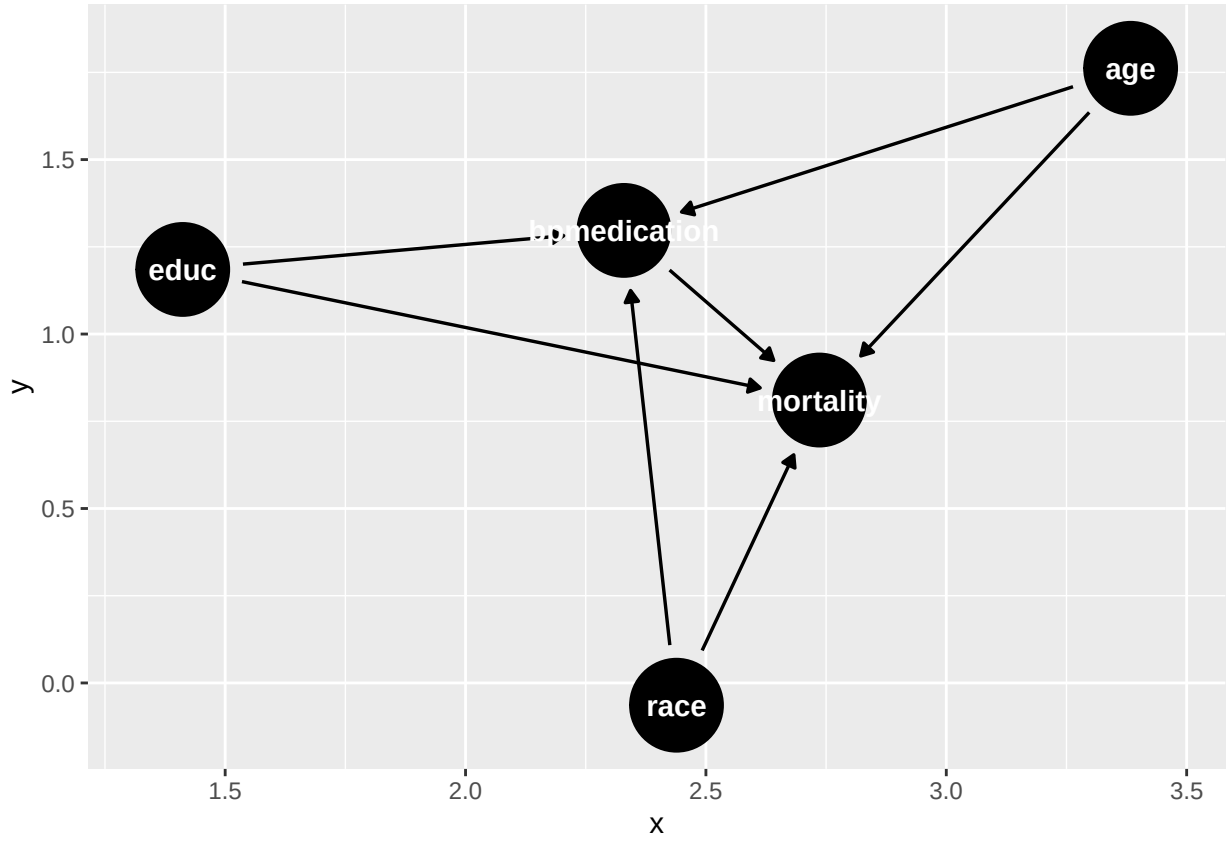
TMLE requires estimating two models:

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

library(ggdag)
dag <- dagify(mortality ~ bpmedication, mortality~ age, bpmedication~age, mortality~ race, bpmedication
ggdag(dag)
```



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.ranger', '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 %>%
  mutate(Y = ifelse(mortality > 0.5,
                        1,
                        0)) %>%
```

```
   dplyr::rename(A = blood_pressure_medication) %>%
  select(Y, A, simplified_race, age, college_educ)

# 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)

# 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:

```
# 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 0.553 0.553 0.251
## 2     0     0 0.567 0.567 0.262
## 3     1     0 0.574 0.574 0.268
## 4     0     0 0.542 0.542 0.243
## 5     0     0 0.568 0.568 0.263
## 6     1     0 0.569 0.569 0.264
```

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

```
## [1] -0.3039574
```

```
#the ATE is -0.3082129
```

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 means that the estimator will be asymptotically unbiased if either 1) the outcome model is correctly specified or 2) the propensity score model is correctly specified. The outcome model relies on theory and content knowledge to identify the estimator and the propensity score model is like matching, where confounding is addressed through identifying controls that were similar to the cases.

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'm so sorry, but I couldn't finish this section. I ran into some trouble with LTMLE and am leaving fo
```

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

## LTMLE estimate
```

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?

I would be worried about time-dependent confounding whereby controlling for it would mean controlling for a mediator for the previous timepoint. I would be less concerned about age than blood pressure as time-dependent confounders because while age is important to consider, I don't think it would be considered a mediator in the same way blood pressure would. For example, age simply increases linearly with time and medication wouldn't affect that rate, where medication could affect people's blood pressure differently. Controlling for blood pressure when examining the impact of medication over time could be partially removing the effects of medication.