Project 8 Template

```
# 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(here('Projects/Project\ 8/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

```
head(heart_disease)
```

```
## # A tibble: 6 x 14
##     age sex_at_birth simplified_race college_educ income_thousands   bmi
##   <dbl>        <dbl>           <dbl>        <dbl>             <dbl> <dbl>
## 1  32.9            0               1            2              91.3  27.1
## 2  53.9            1               1            2              38.8  27.6
## 3  65.3            1               3            2              35.5  27.5
## 4  16.8            1               1            2              93.8  24.9
## 5  56.1            1               1            2              85.7  22.8
## 6  57.2            1               1            2              70.8  24.0
## # i 8 more variables: blood_pressure <dbl>, chol <dbl>,
## #   blood_pressure_medication <dbl>, bmi_2 <dbl>, blood_pressure_2 <dbl>,
## #   chol_2 <dbl>, blood_pressure_medication_2 <dbl>, mortality <dbl>
```

```
heart_disease_t1 <- heart_disease %>%
  select(-ends_with("_2"))
```

```r
head(heart_disease_t1)
```

```
## # A tibble: 6 x 10
##     age sex_at_birth simplified_race college_educ income_thousands   bmi
##   <dbl>        <dbl>           <dbl>        <dbl>             <dbl> <dbl>
## 1  32.9            0               1            2              91.3  27.1
## 2  53.9            1               1            2              38.8  27.6
## 3  65.3            1               3            2              35.5  27.5
## 4  16.8            1               1            2              93.8  24.9
## 5  56.1            1               1            2              85.7  22.8
## 6  57.2            1               1            2              70.8  24.0
## # i 4 more variables: blood_pressure <dbl>, chol <dbl>,
## #   blood_pressure_medication <dbl>, mortality <dbl>
```

```r
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
# Fit SuperLearner Model

## sl lib
## ^ select library after creating train/test split

## Train/Test split
heart_t1_split <- initial_split(heart_disease_t1, prop = 3/4)

heart_t1_train <- training(heart_t1_split)

heart_t1_test <- testing(heart_t1_split)

t1_y_train <- heart_t1_train %>%
  pull(mortality)
```

```r
t1_y_test <- heart_t1_test %>%
  pull(mortality)

t1_x_train <- heart_t1_train %>%
  select(-mortality)

t1_x_test <- heart_t1_test %>%
  select(-mortality)

## Train SuperLearner
set.seed(123)
# Lasso
sl_group <- SuperLearner(Y = t1_y_train,              # target
                         X = t1_x_train,              # features
                         family = binomial(),      # binomial : 1,0s
                         SL.library = c("SL.glmnet",
                                        "SL.mean",
                                        "SL.knn",
                                        "SL.nnls",
                                        "SL.ranger")) # list models as a vector
```

```
## Loading required namespace: ranger
```

```r
## Risk and Coefficient of each model
sl_group
```

```
##
## Call:
## SuperLearner(Y = t1_y_train, X = t1_x_train, family = binomial(), SL.library = c("SL.glmnet",
##     "SL.mean", "SL.knn", "SL.nnls", "SL.ranger"))
##
##
##                        Risk         Coef
## SL.glmnet_All    0.2360582 0.3671443367
## SL.mean_All      0.2496248 0.0000000000
## SL.knn_All       0.2746666 0.0000000000
## SL.nnls_All   7652.0194984 0.0001923577
## SL.ranger_All    0.2322705 0.6326633056
```

```r
## Discrete winner and superlearner ensemble performance
sl_group$cvRisk[which.min(sl_group$cvRisk)]
```

```
## SL.ranger_All
##     0.2322705
```

```r
## Confusion Matrix
```

```r
preds <-
  predict(sl_group,             # use the superlearner not individual models
          t1_x_test,          # prediction on test set
          onlySL = TRUE)  # use only models that were found to be useful (had weights)


# start with y_test
validation <-
  t1_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, # this is the code that transforms the probabilities into binary
                           1,
                           0))
```

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

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

```
## # A tibble: 6 x 2
##     obs  pred
##   <dbl> <dbl>
## 1     1     1
## 2     1     1
## 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  380  175
##          1  858 1087
##
##                Accuracy : 0.5868
##                  95% CI : (0.5672, 0.6062)
##     No Information Rate : 0.5048
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1692
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3069
##             Specificity : 0.8613
##          Pos Pred Value : 0.6847
##          Neg Pred Value : 0.5589
##              Prevalence : 0.4952
##          Detection Rate : 0.1520
##    Detection Prevalence : 0.2220
##       Balanced Accuracy : 0.5841
```

```
##
##        '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)?

The choice of the "best" model in cross-validation can be sensitive to the particular sample of data used. Small changes in the training data can lead to different models being selected. The SuperLearner's ensemble method, by contrast, incorporates model averaging which can provide a more stable and reliable prediction by averaging across these variations. The SuperLearner method also allow for optimization across a variety of loss functions and is not tied to a single metric of success as cross-validation might be. This flexibility means it can be tailored to prioritize different aspects of prediction accuracy based on the specific context or application.

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 $P(A|W)$

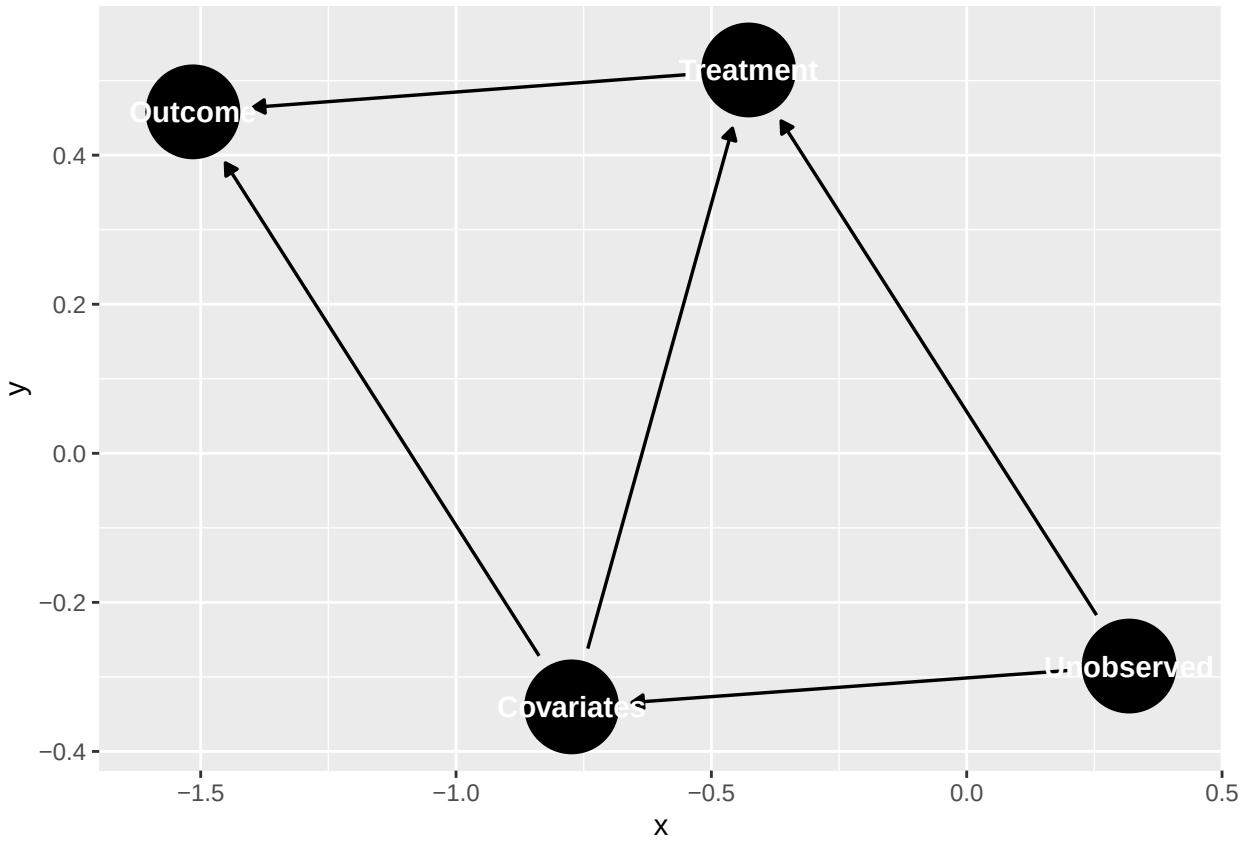
Using ggdag and daggity, draw a directed acylcic graph (DAG) that describes the relationships between the outcome, treatment, and covariates/predictors. Note, if you think there are covariates that are not related to other variables in the dataset, note this by either including them as freestanding nodes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

```
# DAG for TMLE
# install packages

# install.packages("ggdag")
# install.packages("dagitty")
# library(ggdag)
# library(dagitty)
```

```
# Define the simplified DAG
# Define the DAG
dag <- dagitty::dagitty("dag {
  Covariates -> Treatment
  Covariates -> Outcome
  Unobserved -> Covariates
  Unobserved -> Treatment
  Treatment -> Outcome
}")

# Plot the DAG
ggdag_plot <- ggdag::ggdag(dag, text = TRUE)
print(ggdag_plot)
```



The DAG here shows covariates as a single node that are influenced by unobserved variables and themselves influence both treatment and outcome. I did not include any isolate covariates although I'm wondering if t2 treatments could be considered as such for this DAG. They wouldn't influence the outcome we have subset these data for and by definition cannot influence our t1 treatment.

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

```
head(heart_disease_t1)
```

```
## # A tibble: 6 x 10
##     age sex_at_birth simplified_race college_educ income_thousands   bmi
##   <dbl>        <dbl>           <dbl>        <dbl>             <dbl> <dbl>
## 1  32.9            0               1            2              91.3  27.1
## 2  53.9            1               1            2              38.8  27.6
## 3  65.3            1               3            2              35.5  27.5
## 4  16.8            1               1            2              93.8  24.9
## 5  56.1            1               1            2              85.7  22.8
## 6  57.2            1               1            2              70.8  24.0
## # i 4 more variables: blood_pressure <dbl>, chol <dbl>,
```

```
## #   blood_pressure_medication <dbl>, mortality <dbl>
```

```
Y <- heart_disease_t1 %>%
  pull(mortality)

A <- heart_disease_t1$blood_pressure_medication

W <- heart_disease_t1 %>%
  select(-c(blood_pressure_medication, mortality))
```

```
sl_libs <- c('SL.mean', 'SL.ranger', 'SL.glm')
```

```
set.seed(123)

# implement above all in one step using tmle
# ----------
tmle_fit <-
  tmle::tmle(Y = Y,                  # outcome
             A = A,                  # treatment
             W = W,                  # baseline covariates
             Q.SL.library = sl_libs, # libraries for initial estimate
             g.SL.library = sl_libs) # libraries for prob to be in treatment

# view results
tmle_fit
```

```
##  Additive Effect
##    Parameter Estimate:  -0.35505
##    Estimated Variance:  6.0386e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.37028, -0.33982)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.32035
##    Estimated Variance:  0.00014464
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.34393, -0.29678)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.37284
##    Estimated Variance:  5.3265e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.38714, -0.35853)
```

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.

To be doubly robust requires either fitting the right model to estimate the expected outcome correctly or fitting the model to estimate the probability of treatment correctly. Having either one of these will still yield consistent estimations, i.e. the bias of the estimator will tend towards 0 as the sample size increases. If neither condition can be met then causal inference cannot be defined. If the propensity score model, which predicts

the probability of receiving treatment based on covariates, is accurate, it addresses treatment assignment bias. Conversely, if the outcome model, which predicts outcomes based on treatment and covariates, is correct, it compensates for any inaccuracies in the treatment 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

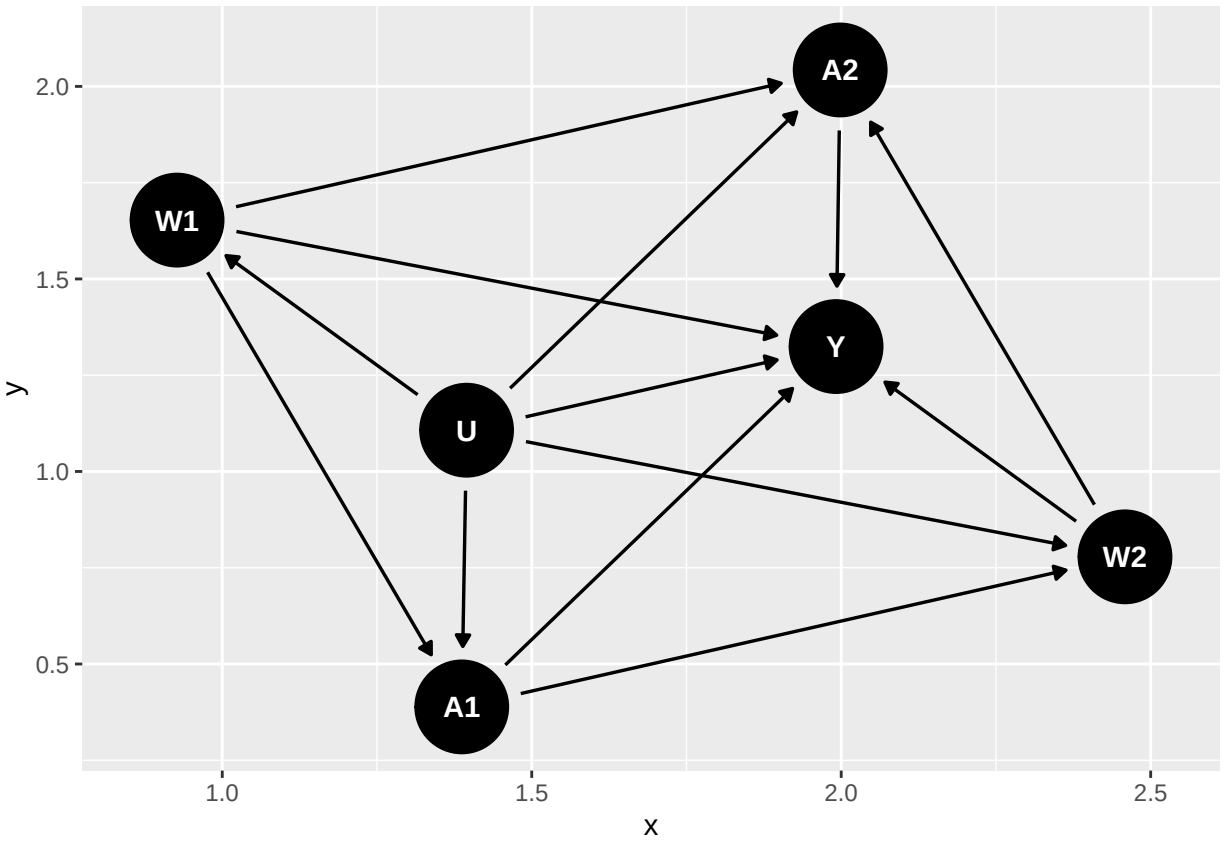
Update your causal diagram to incorporate this new information. **Note**: If your groups divides up sections and someone is working on LTMLE separately from TMLE then just draw a causal diagram even if it does not match the one you specified above.

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
dag <- dagitty::dagitty("dag {
  W1 -> A1 -> W2 -> A2 -> Y
  U -> W1
  U -> W2
  U -> A1
  U -> A2
  U -> Y
  W1 -> Y
  W1 -> A2
  W2 -> Y
  A1 -> Y
}")

# Plot the DAG
ggdag_plot <- ggdag::ggdag(dag, text = TRUE)
print(ggdag_plot)
```



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?

```
library(parallel)
library(SuperLearner)
library(tmle)
library(foreach)

# Setup parallel cluster
num_cores <- parallel::detectCores() - 2
cl <- makeCluster(num_cores)
doParallel::registerDoParallel(cl)
```

```
colnames(heart_disease)
```

```
##  [1] "age"                       "sex_at_birth"
##  [3] "simplified_race"           "college_educ"
##  [5] "income_thousands"          "bmi"
##  [7] "blood_pressure"            "chol"
##  [9] "blood_pressure_medication" "bmi_2"
## [11] "blood_pressure_2"          "chol_2"
## [13] "blood_pressure_medication_2" "mortality"
```

```r
# Randomly sampling a smaller size to hopefully prevent R from crashing
set.seed(123)
heart_disease_small <- heart_disease %>%
  select(c("blood_pressure",
           "blood_pressure_medication",
           "blood_pressure_2",
           "blood_pressure_medication_2",
           "mortality"))

heart_disease_small <- sample_frac(heart_disease_small, size = 0.10)

heart_disease_small_t1 <- heart_disease_small %>%
  select(-ends_with("_2"))
```

```r
## Naive Model (no time-dependent confounding) estimate
# implement tmle
# ----------
result <- ltmle(heart_disease_small_t1,
                Anodes="blood_pressure_medication",
                Lnodes=NULL,      # no Lnodes
                Ynodes="mortality",
                abar=1,
                SL.library=sl_libs)
```

```
## Qform not specified, using defaults:
## formula for mortality:
## Q.kplus1 ~ blood_pressure + blood_pressure_medication
##
## gform not specified, using defaults:
## formula for blood_pressure_medication:
## blood_pressure_medication ~ blood_pressure
##
## Estimate of time to completion: < 1 minute
```

```r
# view
result
```

```
## Call:
## ltmle(data = heart_disease_small_t1, Anodes = "blood_pressure_medication",
##     Lnodes = NULL, Ynodes = "mortality", abar = 1, SL.library = sl_libs)
##
## TMLE Estimate:  0.2738576
```

```r
colnames(heart_disease_small)
```

```
## [1] "blood_pressure"              "blood_pressure_medication"
## [3] "blood_pressure_2"            "blood_pressure_medication_2"
## [5] "mortality"
```

```r
## LTMLE estimate

result_ltmle <- ltmle(heart_disease_small,
```

```r
    Anodes=c("blood_pressure_medication", "blood_pressure_medication_2"),  # two treatment variables
    Lnodes=c("blood_pressure", "blood_pressure_2"),            # L indicator
    Ynodes="mortality",            # outcome
    abar=c(1, 1),          # treatment indicator in Anodes vector
    SL.library = sl_libs)
```

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

## formula for blood_pressure_2:

## Q.kplus1 ~ blood_pressure + blood_pressure_medication

## formula for mortality:

## Q.kplus1 ~ blood_pressure + blood_pressure_medication + blood_pressure_2 +     blood_pressure_medica

##

## gform not specified, using defaults:

## formula for blood_pressure_medication:

## blood_pressure_medication ~ blood_pressure

## formula for blood_pressure_medication_2:

## blood_pressure_medication_2 ~ blood_pressure + blood_pressure_medication +     blood_pressure_2

##

## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds
## Error in pred[, "1"] : subscript out of bounds

## Estimate of time to completion: < 1 minute

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
```

```
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Error in pred[, "1"] : subscript out of bounds
## Warning in FUN(X[[i]], ...): Error in algorithm SL.ranger  on full data
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)
## Warning in SuperLearner::SuperLearner(Y = Y.subset, X = X.subset, SL.library =
## SL.library, : Coefficients already 0 for all failed algorithm(s)
```

```
result_ltmle
```

```
## Call:
## ltmle(data = heart_disease_small, Anodes = c("blood_pressure_medication",
##     "blood_pressure_medication_2"), Lnodes = c("blood_pressure",
##     "blood_pressure_2"), Ynodes = "mortality", abar = c(1, 1),
##     SL.library = sl_libs)
##
## TMLE Estimate:  0.2852149
```

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 occurs when covariates affected by prior treatments influence future treatments and outcomes. For example, if blood pressure is lowered by medication and then influences further treatment decisions, it acts as a time-dependent confounder. Age, while continuously changing, typically does not change due to interventions and thus doesn't confound in the same way; however, its interaction with other variables and treatments can affect outcomes and should be controlled for.