Project 8 Template

```
# Add to this package list for additional SL algorithms

set.seed(5)

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

# Set the working directory to where the file is located
setwd("/Users/jr/Downloads")

# Read the CSV file into a data frame
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.
```

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

## Train SuperLearner

## Risk and Coefficient of each model
```

```r
## Discrete winner and superlearner ensemble performance

## Confusion Matrix
```

```r
#Initiatl Split

hd_split <-
 initial_split(heart_disease, prop = 3/4)

#Train Split

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


y_train <- train %>% pull(mortality)

x_train <-
  train %>%
  # drop the target variable
  select(-mortality, -blood_pressure_2, -chol_2)

x_train_LTMLE <-
  train %>%
  # drop the target variable
  select(-mortality)

# Test Plit
test <-
  # Declare the training set with rsample::training()
  testing(hd_split)

  y_test <- test %>% pull(mortality)

  x_test <-
  test %>%
  # drop the target variable
  select(-mortality, -blood_pressure_2, -chol_2)

x_test_LTMLE <-
  test %>%
  # drop the target variable
  select(-mortality)
```

```r
set.seed(10)

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',
```

```
                                  'SL.xgboost',
                                  'SL.knn'))
```

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

```
## Loading required namespace: xgboost
```

```
sl
```

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = c("SL.mean",
##     "SL.glmnet", "SL.ranger", "SL.xgboost", "SL.knn"))
##
##
##                       Risk      Coef
## SL.mean_All     0.2499200 0.0000000
## SL.glmnet_All   0.2368078 0.3838025
## SL.ranger_All   0.2331234 0.6161975
## SL.xgboost_All 0.2504821 0.0000000
## SL.knn_All      0.2701572 0.0000000
```

```
preds <-
  predict(sl,             # use the superlearner not individual models
          x_test,         # prediction on test set
          onlySL = TRUE)

# start with y_test
validation <-
  y_test %>%
  # add our predictions - first column of predictions. This is not a df so it is calling a vector.
  bind_cols(preds$pred[,1]) %>%
  # rename columns
  rename(obs = `...1`,      # actual observations
         pred = `...2`) %>% # predicted prob
  # We are renaming them based on how we bounded them together.
  # change pred column so that obs above .5 are 1, otherwise 0
  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     0     1
## 3     0     0
## 4     0     1
## 5     1     1
## 6     0     1
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  420  240
##          1  753 1087
##
##                Accuracy : 0.6028
##                  95% CI : (0.5833, 0.6221)
##     No Information Rate : 0.5308
##     P-Value [Acc > NIR] : 2.465e-13
##
##                   Kappa : 0.1818
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3581
##             Specificity : 0.8191
##          Pos Pred Value : 0.6364
##          Neg Pred Value : 0.5908
##              Prevalence : 0.4692
##          Detection Rate : 0.1680
##    Detection Prevalence : 0.2640
##       Balanced Accuracy : 0.5886
##
##        'Positive' Class : 0
##
```

```r
# Load the caret package
library(caret)

# Assuming 'validation' is your dataset with 'pred' as predictions and 'obs' as actual observations
cm <- confusionMatrix(as.factor(validation$pred), as.factor(validation$obs))

# Print the confusion matrix
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  420  240
##          1  753 1087
##
##                Accuracy : 0.6028
##                  95% CI : (0.5833, 0.6221)
##     No Information Rate : 0.5308
##     P-Value [Acc > NIR] : 2.465e-13
##
##                   Kappa : 0.1818
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.3581
##             Specificity : 0.8191
##          Pos Pred Value : 0.6364
##          Neg Pred Value : 0.5908
##              Prevalence : 0.4692
##          Detection Rate : 0.1680
##    Detection Prevalence : 0.2640
##       Balanced Accuracy : 0.5886
##
##        'Positive' Class : 0
##
```

```r
# Recall (Sensitivity) for each class
recall <- cm$byClass['Sensitivity']

# Precision (Positive Predictive Value) for each class
precision <- cm$byClass['Positive Predictive Value']

# Print recall and precision
print(paste("Recall:", recall))
```

```
## [1] "Recall: 0.358056265984655"
```

```r
print(paste("Precision:", precision))
```

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

```r
conf_matrix <- cm$table

# Access elements from the confusion matrix
true_negatives <- conf_matrix[1, 1]  # TN
false_positives <- conf_matrix[1, 2]  # FP
false_negatives <- conf_matrix[2, 1]  # FN
true_positives <- conf_matrix[2, 2]  # TP

# Print values
cat("True Negatives (TN):", true_negatives, "\n")
```

```
## True Negatives (TN): 420
```

```r
cat("False Positives (FP):", false_positives, "\n")
```

```
## False Positives (FP): 240
```

```r
cat("False Negatives (FN):", false_negatives, "\n")
```

```
## False Negatives (FN): 753
```

```r
cat("True Positives (TP):", true_positives, "\n")
```

```
## True Positives (TP): 1087
```

```r
# Calculate precision
precision <- true_positives / (true_positives + false_positives)

# Print the precision
print(paste("Precision:", precision))
```

```
## [1] "Precision: 0.819140919366993"
```

```
# Calculate recall
recall <- true_positives / (true_positives + false_negatives)

# Print the recall
print(paste("Recall:", recall))
```

```
## [1] "Recall: 0.590760869565217"
```

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

Ensemble SuplerLearners are preferable for several reasons. First we can minimize over fitting our data by meaning out idiosyncrasies to the data through several iterations of models. Furthermore by using ensemble method the super learner can utilize aspects that respective individual models perform well on. We could for example have some models that do well as reducing variance while another models does particularly well accounting for outliers while another can accommodate for all functional forms. Also as our data scales we may also include more models to capture the true relationship of our 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.

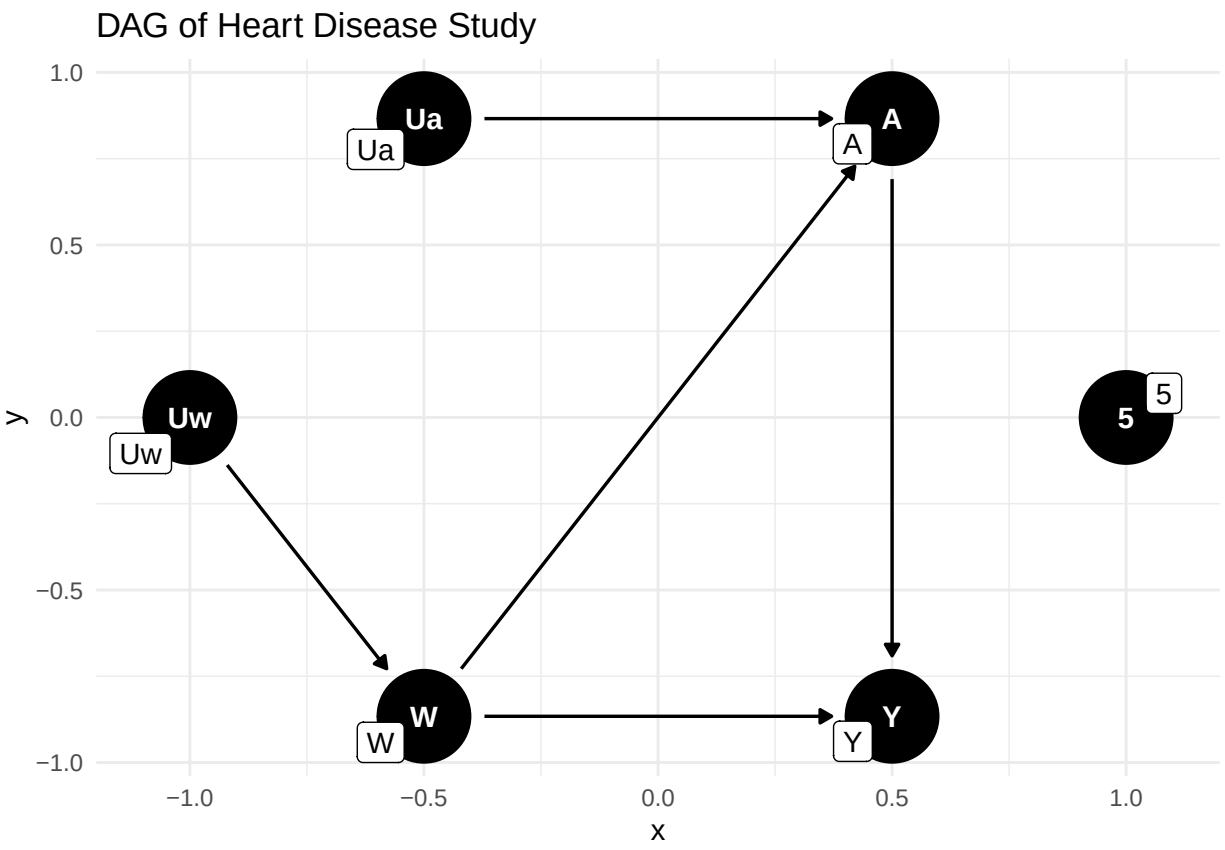
```
# DAG for TMLE

#
library(ggdag)
library(dagitty)
# Define DAG

dag <- dagitty::dagitty("
dag {
Uw [unobserved]
Ua [unobserved]
5
W -> A -> Y
W -> Y
Uw -> W
Ua -> A
}")
```

```
ggdag <- ggdag::ggdag(dag, text = TRUE, use_labels = "name", layout = "circle") +
theme_minimal() +
ggtitle("DAG of Heart Disease Study")

print(ggdag)
```



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

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


Y <-
  heart_disease %>%
```

```
  pull(mortality)

W <- heart_disease %>% select(sex_at_birth, age, simplified_race,college_educ, income_thousands, bmi, bl

W_A<- heart_disease %>%
  rename(A= blood_pressure_medication) %>%
  select(A)

A <- W_A$A
```

```
set.seed(10)

# 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:  -0.3542
##    Estimated Variance:  6.3729e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.36985, -0.33855)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.32006
##    Estimated Variance:  0.00014517
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.34368, -0.29645)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.37109
##    Estimated Variance:  5.639e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.3858, -0.35637)
```

```
summary(tmle_fit)
```

```
##  Initial estimation of Q
##   Procedure: cv-SuperLearner, ensemble
##   Model:
##       Y ~  SL.mean_All + SL.glmnet_All + SL.ranger_All + SL.xgboost_All + SL.glm_All
##
##   Coefficients:
##       SL.mean_All    0
##     SL.glmnet_All    0.2977491
##     SL.ranger_All    0.6770609
##    SL.xgboost_All    0.02518995
##        SL.glm_All    0
```

```
##
##   Cross-validated R squared :  0.0839
##
##  Estimation of g (treatment mechanism)
##   Procedure: SuperLearner, ensemble
##   Model:
##       A ~  SL.mean_All + SL.glmnet_All + SL.ranger_All + SL.xgboost_All + SL.glm_All
##
##   Coefficients:
##       SL.mean_All    0.04184357
##     SL.glmnet_All    0.04158448
##     SL.ranger_All    0.1571726
##    SL.xgboost_All    0.04714281
##        SL.glm_All    0.7122566
##
##  Estimation of g.Z (intermediate variable assignment mechanism)
##   Procedure: No intermediate variable
##
##  Estimation of g.Delta (missingness mechanism)
##   Procedure: No missingness, ensemble
##
##  Bounds on g: (0.0054, 1)
##
##  Bounds on g for ATT/ATE: (0.0054, 0.9946)
##
##  Additive Effect
##    Parameter Estimate:  -0.3542
##    Estimated Variance:  6.3729e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.36985, -0.33855)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.32006
##    Estimated Variance:  0.00014517
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.34368, -0.29645)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.37109
##    Estimated Variance:  5.639e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.3858, -0.35637)
```

```
print(paste("Estimated Average Treatment Effect (ATE) :", tmle_fit$estimates$ATE))
```

```
## [1] "Estimated Average Treatment Effect (ATE) : -0.354200897674096"
## [2] "Estimated Average Treatment Effect (ATE) : 6.3728703653553e-05"
## [3] "Estimated Average Treatment Effect (ATE) : c(-0.36984734106489, -0.338554454283301)"
## [4] "Estimated Average Treatment Effect (ATE) : 0"
## [5] "Estimated Average Treatment Effect (ATE) : NA"
## [6] "Estimated Average Treatment Effect (ATE) : c(NA, NA)"
## [7] "Estimated Average Treatment Effect (ATE) : c(-Inf, NA)"
## [8] "Estimated Average Treatment Effect (ATE) : c(NA, Inf)"
```

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

Double robustness estimators is an estimator where we can ensure our model is specified correctly by either having the correct functional form of the relationship between main predictors and our outcome variable including potential nonlinearities, interactions etc. Whereas the propensity based model we identify the correct set of covariates that are predictive of receiving the treatment. If this model accurately represents the true propensity score, then it helps in properly adjusting for confounding, ensuring that the treatment and control groups are comparable on all observed covariates. By using this we essentially double our chances of having an unbiased estimate where as in other cases we such as matching we can only obtain an unbiased treatment effect when propensity scores are accurate with enough comparable groups. In effect even if we mispecifie one model in the context of double robustness if the alternative model is specified correctly we can rely on that 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.

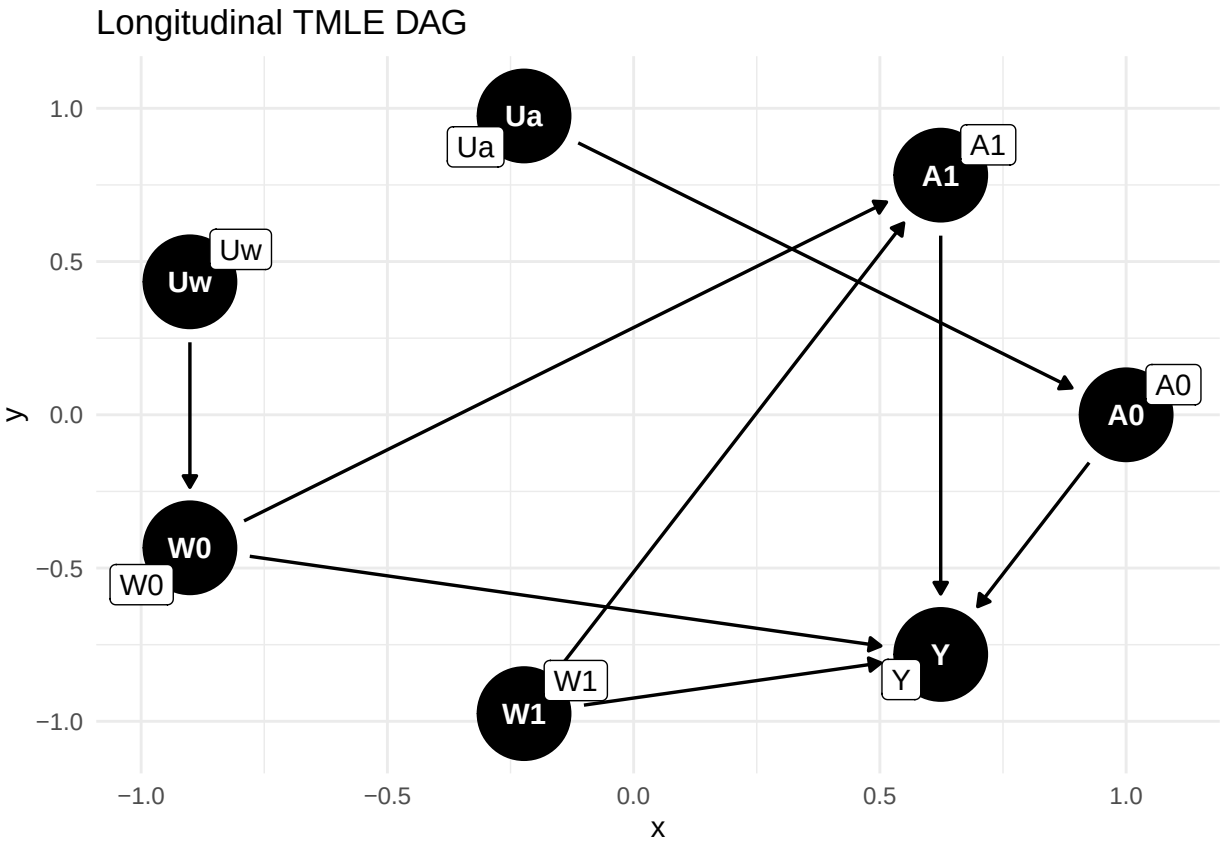
Hint: Keep in mind that any of the variables that end in "_2" are likely affected by both the previous covariates and the first treatment when drawing your DAG.

```
# DAG for TMLE

dag <- dagitty::dagitty("
dag {
Uw [unobserved]
Ua [unobserved]
W1-> A1 -> Y
W1 -> Y
W0 -> Y
W0 -> A1
A0 -> Y
Uw -> W0
Ua -> A0
}"
```

```
)
ggdag <- ggdag::ggdag(dag, text = TRUE, use_labels = "name", layout = "circle") +
theme_minimal() +
ggtitle("Longitudinal TMLE DAG")

# Plot DAG
print(ggdag)
```



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?

Answer

We should see a a less bias estiamte using the LTMLE that accounts for time dependent confounding but what direction or degree of change would require running the model in full. However we should have confidence that there would be an increase in precision.

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

## LTMLE estimate
set.seed(123)  # Set a random seed for reproducibility
subset_df <- heart_disease %>%
  sample_n(size = 5000)
```

```r
data_obs_ltmle <-
  heart_disease %>%
  # need to specify W1, W2, etc
  rename(Y = mortality, A = blood_pressure_medication, W1 = age, W2= sex_at_birth, W3 = simplified_race
  select(W1, W2, W3, W4, W5, W6, W7, W8, A, Y)

result_ltmle_uno <- ltmle(data_obs_ltmle, # 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
```

```r
# view
result_ltmle_uno
```

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

```r
sl_libs_2 <- c('SL.mean',
               'SL.glmnet',
               'SL.ranger',
               'SL.glm')
```

```r
#ltmle(subset_df,
#      Anodes=c("blood_pressure_medication", "blood_pressure_medication_2"),  # two treatment variables
 #     Lnodes=c("bmi", "blood_pressure", "chol", "bmi_2", "blood_pressure_2", "chol_2"),            # L
#      Ynodes="mortality",            # outcome
#      abar=c(1, 1),          # treatment indicator in Anodes vector
#      SL.library = sl_libs_2)


# I could not get the time counfounding LTMLE to work. In previous attempts I attempted to

# 1.partion the function across cores . 2. Reduce the sample size by half and remove xgboost. 3. Reduce
```

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?

Answer

No we would should not consider age and blood pressure similarly. Blood pressure measured at multiple time points is a prime example of a time-varying confounder that may also act as a mediator in the pathway between the treatment (e.g., a cardiovascular drug) and the outcome (e.g., heart attack or mortality). Age, as a running variable, increases uniformly for all subjects as time progresses. It does not change in response to treatment, but it may still affect the likelihood of receiving treatment and the risk of the outcome. Although age changes over time, it is not typically affected by treatment. It acts as a confounder due to its association with both the likelihood of receiving treatment and the risk of outcomes but does not mediate the effect of treatment on the outcome. Age must be adjusted for in any longitudinal analysis, but the method of adjustment can be more straightforward compared to mediators like blood pressure.