Project 8 Template

```r
# Add to this package list for additional SL algorithms
pacman::p_load(
  tidyverse,
  ggthemes,
  ltmle,
  tmle,
  SuperLearner,
  tidymodels,
  caret,
  dagitty,
  ggdag,
  arm,
  xgboost,
  here,
  ggplot2,
  dplyr)

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

```r
# Fit SuperLearner Model

# Prepare data for SuperLearner.

## Drop the variables that have _2

heart_disease_sl <- heart_disease %>%
  dplyr :: select(age, sex_at_birth, simplified_race, college_educ, income_thousands, bmi,
         blood_pressure, chol, blood_pressure_medication, mortality)
glimpse(heart_disease_sl)
```

```
## Rows: 10,000
## Columns: 10
## $ age                       <dbl> 32.92951, 53.92344, 65.33631, 16.82682, 56.0~
## $ sex_at_birth              <dbl> 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,~
## $ simplified_race           <dbl> 1, 1, 3, 1, 1, 1, 1, 1, 3, 3, 3, 2, 1, 1, 1,~
## $ college_educ              <dbl> 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 2, 2, 2, 2,~
## $ income_thousands          <dbl> 91.32660, 38.76890, 35.48435, 93.77726, 85.7~
## $ bmi                       <dbl> 27.09986, 27.61615, 27.52499, 24.88569, 22.7~
## $ blood_pressure            <dbl> 146.9862, 125.7578, 131.6789, 131.2926, 130.~
## $ chol                      <dbl> 211.4630, 187.8418, 197.0794, 229.6352, 208.~
## $ blood_pressure_medication <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,~
## $ mortality                 <dbl> 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,~
```

```r
## sl lib

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
# Will use the following 5 algorithms: 1) SL.mean (to serve as a baseline), 2) SL.glmnet,
# 3) SL.randomForest, 4) SL.xgboost, 5) SL.bayesglm
SL.library <- c("SL.mean", "SL.glmnet", "SL.randomForest", "SL.xgboost", "SL.bayesglm")

## Train/Test split

heart_disease_sl_split <-
  initial_split(heart_disease_sl, prop = 3/4) # create initial split

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

# y_train
y_train <-
  train %>%
  # pull and save as vector
  pull(mortality)

# x_train
x_train <-
  train %>%
  # drop the target variable
  dplyr :: select(-mortality)

# Testing
# ----------
test <-
  testing(heart_disease_sl_split)

# y test
y_test <-
  test %>%
  pull(mortality)

# x test
x_test <-
  test %>%
  dplyr :: select(-mortality)

## Train SuperLearner

# set seed
set.seed(987)

# multiple models
# ----------
sl = SuperLearner(Y = y_train,
                  X = x_train,
                  family = binomial(),
                  SL.library = SL.library)
```

```
## Loading required namespace: randomForest
```

```
## Risk and Coefficient of each model

sl$cvRisk
```

```
##         SL.mean_All       SL.glmnet_All SL.randomForest_All      SL.xgboost_All
##           0.2497740           0.2368560           0.2319627           0.2490622
##     SL.bayesglm_All
##           0.2370547
```

```
sl$coef
```

```
##         SL.mean_All       SL.glmnet_All SL.randomForest_All      SL.xgboost_All
##            0.000000            0.349424            0.650576            0.000000
##     SL.bayesglm_All
##            0.000000
```

```
## Discrete winner and superlearner ensemble performance

discrete_winner <- which.min(sl$cvRisk)
winner_name <- SL.library[discrete_winner]

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


# start with y_test
validation <-
  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,
                            1,
                            0))
```

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

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

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

```
## Confusion Matrix

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  436  242
##          1  766 1056
##
##                Accuracy : 0.5968
##                  95% CI : (0.5773, 0.6161)
##     No Information Rate : 0.5192
##     P-Value [Acc > NIR] : 3.729e-15
##
##                   Kappa : 0.1792
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3627
##             Specificity : 0.8136
##          Pos Pred Value : 0.6431
##          Neg Pred Value : 0.5796
##              Prevalence : 0.4808
##          Detection Rate : 0.1744
##    Detection Prevalence : 0.2712
##       Balanced Accuracy : 0.5881
##
##        'Positive' Class : 0
##
```

```
# Overall accuracy: 0.586
# Overall recall (aka sensistivity): 0.3598
# Overall precision (aka positive predictive value):  0.5955
```

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)?

SuperLearner allows us to combine algorithms, which each pick up on different relationships. Doing this can help us reduce model bias and variance at the same time. On the other hand using the "best" single algorithm (defined by minimizing risk) could mean taht we choose an algorithm that had the lowest CV risk by chance, especially if risks are close. Choosing the single best model may also overfit to the validation set. SuperLearner reduces this by blending across models rather than fully committing to one.

Targeted Maximum Likelihood Estimation

Causal Diagram

TMLE requires estimating two models:

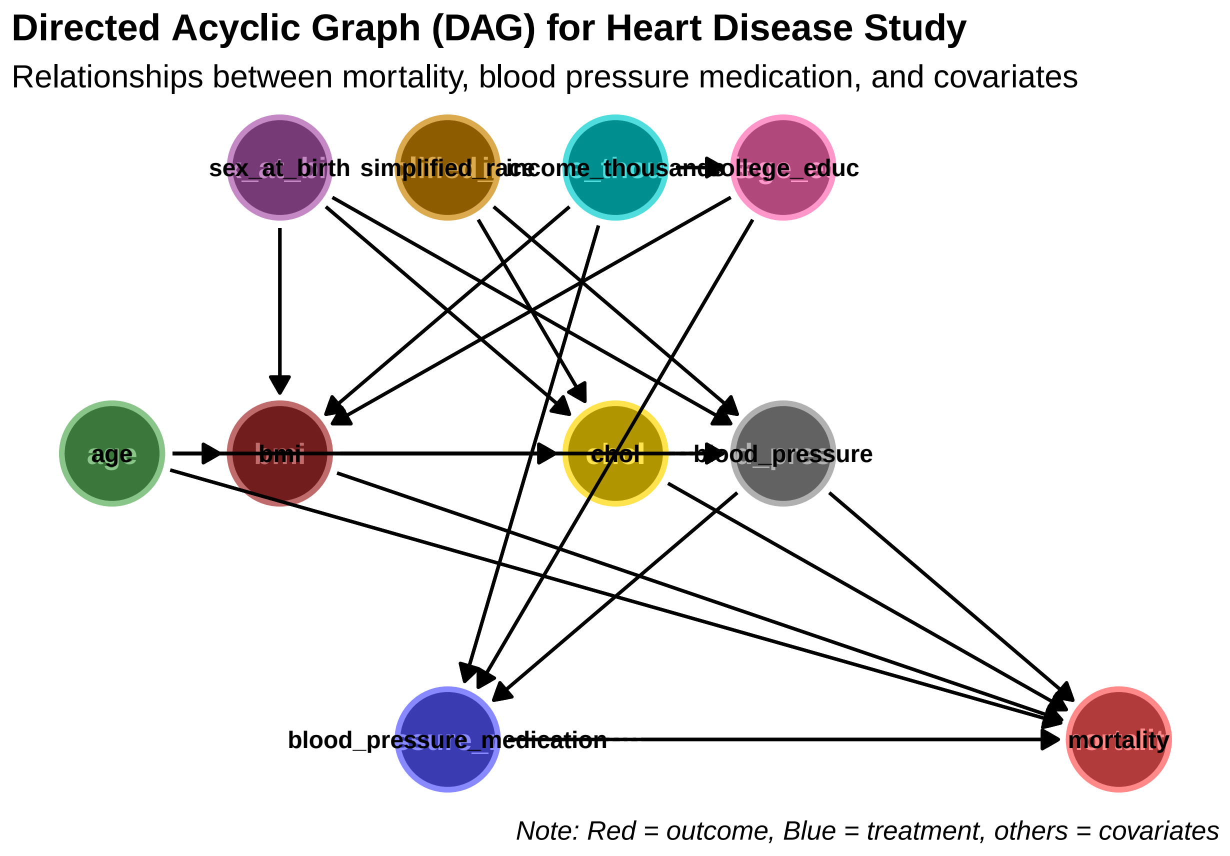
1. The outcome model, or the relationship between the outcome and the treatment/predictors, $P(Y|(A, W)$.
2. The propensity score model, or the relationship between assignment to treatment and predictors $P(A|W)$

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.

Cindy note: I think all covariates are related to other variables in the dataset, which is reflected in the DAG below. I also use adjustmentSets() to help me identify variable sets that I can condition on to block non-causal paths without blocking the causal effect (given my covariates).

```
# DAG for TMLE
heart_dag <- dagitty('dag {
  mortality [outcome, pos="6,0"]
  blood_pressure_medication [exposure, pos="2,0"]
  age [pos="0,1"]
  sex_at_birth [pos="1,2"]
  simplified_race [pos="2,2"]
  income_thousands [pos="3,2"]
  college_educ [pos="4,2"]
  bmi [pos="1,1"]
  chol [pos="3,1"]
  blood_pressure [pos="4,1"]

  age -> bmi
  age -> blood_pressure
  age -> chol
  age -> mortality

  sex_at_birth -> bmi
  sex_at_birth -> blood_pressure
  sex_at_birth -> chol

  simplified_race -> blood_pressure
  simplified_race -> chol
```

```r
  income_thousands -> college_educ
  income_thousands -> bmi

  college_educ -> bmi

  bmi -> blood_pressure
  bmi -> chol
  bmi -> mortality

  chol -> blood_pressure
  chol -> mortality

  blood_pressure -> blood_pressure_medication
  blood_pressure -> mortality

  blood_pressure_medication -> mortality

  income_thousands -> blood_pressure_medication
  college_educ -> blood_pressure_medication
}')

#Plot the DAG

heart_dag_tidy <- tidy_dagitty(heart_dag)
dag_plot <- ggdag(heart_dag_tidy, layout = "auto") +
  theme_dag() +
  # Nodes with better visibility
  geom_dag_point(aes(color = name), size = 18, alpha = 0.7) +
  # Text styling: BLACK text only
  geom_dag_text(size = 3,
                color = "black",
                fontface = "bold",
                family = "sans") +  # Use sans-serif font for clarity
  # Improved edges
  geom_dag_edges(aes(),
               edge_width = 0.6,
               arrow_directed = grid::arrow(length = grid::unit(7, "pt"), type = "closed")) +
  scale_color_manual(values = c(
    "mortality" = "#FF5555",
    "blood_pressure_medication" = "#5555FF",
    "age" = "#55AA55",
    "sex_at_birth" = "#AA55AA",
    "simplified_race" = "#CD8500",
    "income_thousands" = "#00CDCD",
    "college_educ" = "#FF69B4",
    "bmi" = "#A52A2A",
    "chol" = "#FFD700",
    "blood_pressure" = "#8E8E8E"
  )) +
  labs(title = "Directed Acyclic Graph (DAG) for Heart Disease Study",
       subtitle = "Relationships between mortality, blood pressure medication, and covariates",
       caption = "Note: Red = outcome, Blue = treatment, others = covariates") +
  theme(
    legend.position = "none",
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 12),
    plot.caption = element_text(face = "italic")
  )

# Display the plot
print(dag_plot)
```



Directed Acyclic Graph (DAG) for Heart Disease Study
Relationships between mortality, blood pressure medication, and covariates
sex_at_birth
simplified_race
income_thousands
college_educ
age
bmi
chol
blood_pressure
blood_pressure_medication
mortality
Note: Red = outcome, Blue = treatment, others = covariates

```
# Identify adjustment sets for estimating the causal effect of treatment on outcome
adjustment_sets <- adjustmentSets(heart_dag, exposure = "blood_pressure_medication", outcome = "mortali
print("Minimal adjustment sets for estimating the effect of blood pressure medication on mortality:")
```

```
## [1] "Minimal adjustment sets for estimating the effect of blood pressure medication on mortality:"
```

```
print(adjustment_sets)
```

```
## { age, blood_pressure, bmi, chol }
## { age, blood_pressure, bmi, sex_at_birth }
## { blood_pressure, college_educ, income_thousands }
```

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

AdjustemtnSets above allowed me to identify sets of variables that if I condition/adjust for, would block the non-causal paths without blocking the causal effect. Below I use TMLE and the following minimal adjustment set: age, blood_pressure, bmi, and chol. Note that this took a long time to run on my computer so I was unable to run the TMLE using the other two sets. However, I would have liked to compare the results from the three sets if I had enough time/computational power.

```
# Above, I identified three possible minimal adjustment sets:
# 1. { age, blood_pressure, bmi, chol }
# 2. { age, blood_pressure, bmi, sex_at_birth }
# 3. { blood_pressure, college_educ, income_thousands }
# I'll use the first set for this analysis

# use heart_disease_sl
heart_disease_tmle <- heart_disease_sl

# Examine the structure
str(heart_disease_tmle)
```

```
## tibble [10,000 x 10] (S3: tbl_df/tbl/data.frame)
##  $ age                     : num [1:10000] 32.9 53.9 65.3 16.8 56.1 ...
##  $ sex_at_birth            : num [1:10000] 0 1 1 1 1 1 1 1 1 0 ...
##  $ simplified_race         : num [1:10000] 1 1 3 1 1 1 1 1 3 3 ...
##  $ college_educ            : num [1:10000] 2 2 2 2 2 2 1 1 2 1 ...
##  $ income_thousands        : num [1:10000] 91.3 38.8 35.5 93.8 85.7 ...
##  $ bmi                     : num [1:10000] 27.1 27.6 27.5 24.9 22.8 ...
##  $ blood_pressure          : num [1:10000] 147 126 132 131 131 ...
##  $ chol                    : num [1:10000] 211 188 197 230 208 ...
##  $ blood_pressure_medication: num [1:10000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ mortality               : num [1:10000] 1 0 1 0 0 1 1 1 1 1 ...
```

```
# Make sure variables are properly formatted
heart_disease_tmle <- heart_disease_tmle %>%
  mutate(
    # Ensure binary variables are 0/1
    mortality = as.numeric(mortality),
    blood_pressure_medication = as.numeric(blood_pressure_medication),
    sex_at_birth = as.numeric(sex_at_birth),
    college_educ = as.numeric(college_educ),
    # Ensure simplified_race is a factor
    simplified_race = as.factor(simplified_race)
  )

# Define the SuperLearner library (same as above)
SL.library <- c("SL.mean", "SL.glmnet", "SL.randomForest", "SL.xgboost", "SL.bayesglm")

# Extract variables for TMLE
```

```r
Y <- heart_disease_tmle$mortality  # Outcome
A <- heart_disease_tmle$blood_pressure_medication  # Treatment

# Extract covariates based on the first minimal adjustment set from our DAG
# { age, blood_pressure, bmi, chol }
W <- heart_disease_tmle %>%
  dplyr:: select(age, blood_pressure, bmi, chol)

# Double-check the data
summary(Y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5168  1.0000  1.0000
```

```r
summary(A)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1551  0.0000  1.0000
```

```r
summary(W)
```

```
##       age          blood_pressure       bmi              chol
##  Min.   :  1.972   Min.   :106.5   Min.   :14.81   Min.   :151.8
##  1st Qu.: 40.655   1st Qu.:125.1   1st Qu.:23.83   1st Qu.:193.3
##  Median : 50.066   Median :130.1   Median :25.78   Median :203.2
##  Mean   : 50.086   Mean   :132.1   Mean   :26.02   Mean   :203.8
##  3rd Qu.: 59.471   3rd Qu.:136.2   3rd Qu.:27.94   3rd Qu.:213.4
##  Max.   :101.168   Max.   :182.6   Max.   :42.30   Max.   :272.9
```

```r
# Run TMLE
set.seed(42)  # For reproducibility

tmle_result <- tmle(
  Y = Y,  # Outcome: mortality
  A = A,  # Treatment: blood pressure medication
  W = W,  # Covariates: from minimal adjustment set
  Q.SL.library = SL.library,  # SuperLearner library for outcome model
  g.SL.library = SL.library,  # SuperLearner library for propensity score model
  family = "binomial"  # Binary outcome
)

# Print TMLE results
print(tmle_result)
```

```
##  Marginal mean under treatment (EY1)
##    Parameter Estimate:  0.20987
##    Estimated Variance:  1.6299e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.20196, 0.21779)
##
##  Marginal mean under comparator (EY0)
##    Parameter Estimate:  0.56579
##    Estimated Variance:  2.673e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.55566, 0.57592)
##
##  Additive Effect
##    Parameter Estimate:  -0.35592
##    Estimated Variance:  4.3147e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.36879, -0.34304)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.74625
##    Estimated Variance:  2.4261e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.7559, -0.73659)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  0.22944
##    Estimated Variance:  3.3079e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.21817, 0.24071)
##
##  Relative Risk
##    Parameter  Estimate: 0.37094
##    Variance(log scale): 0.00045453
##                p-value: <2e-16
##      95% Conf Interval: (0.35576, 0.38677)
##
##  Odds Ratio
##    Parameter  Estimate: 0.20385
##    Variance(log scale): 0.0010385
##                p-value: <2e-16
##      95% Conf Interval: (0.19137, 0.21714)
```

```r
# Extract the ATE
ate <- tmle_result$estimates$ATE
ate_ci <- c(tmle_result$estimates$ATE$CI[1], tmle_result$estimates$ATE$CI[2])
ate_pvalue <- tmle_result$estimates$ATE$pvalue

# Print the key results
cat("Average Treatment Effect (ATE):", round(ate$psi, 4), "\n")
```

```
## Average Treatment Effect (ATE): -0.3559
```

```r
cat("95% Confidence Interval:", round(ate_ci[1], 4), "to", round(ate_ci[2], 4), "\n")
```

```
## 95% Confidence Interval: -0.3688 to -0.343
```

```r
cat("P-value:", round(ate_pvalue, 4), "\n")
```

```
## P-value: 0
```

```r
# Interpretation based on the sign of the ATE
if (ate$psi < 0) {
  cat("The negative ATE suggests that blood pressure medication reduces mortality risk.\n")
} else if (ate$psi > 0) {
  cat("The positive ATE suggests that blood pressure medication increases mortality risk.\n")
} else {
  cat("The ATE is effectively zero, suggesting no effect of blood pressure medication on mortality.\n")
}
```

```
## The negative ATE suggests that blood pressure medication reduces mortality risk.
```

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.

Double Robust Estimators

A double robust estimator combines an outcome regression model, which models the relationship between covariates, treatment, and outcome, with a propensity score model, which models the probability of treatment assignment given covariates. Double robust methods (like TMLE) use both models in a way that gives us “two chances” to get the right answer. When we use an outcome model approach, we need to correctly specify the outcome model to get consistent estimates. When we use a pure propensity score approach like inverse probability weighting, we need to correctly specify the propensity model. Double robust estimators combine these approaches mathematically so that if either the outcome model is correct or the propensity model is correct, the estimator is consistent.

Traditional statistics courses emphasized theory-driven outcome modeling, focusing on getting the functional form right. Methods like matching instead focus on balancing covariates between treatment groups (like propensity score approaches). Double robust methods give us the best of both worlds because we can leverage our theoretical knowledge of outcome processes (as I did in the DAG step above) while also using propensity scores to account for selection bias. Instead of having to choose between these paradigms and being completely wrong if our choice fails, double robust methods allow us to be partially right and still get consistent estimates.

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.

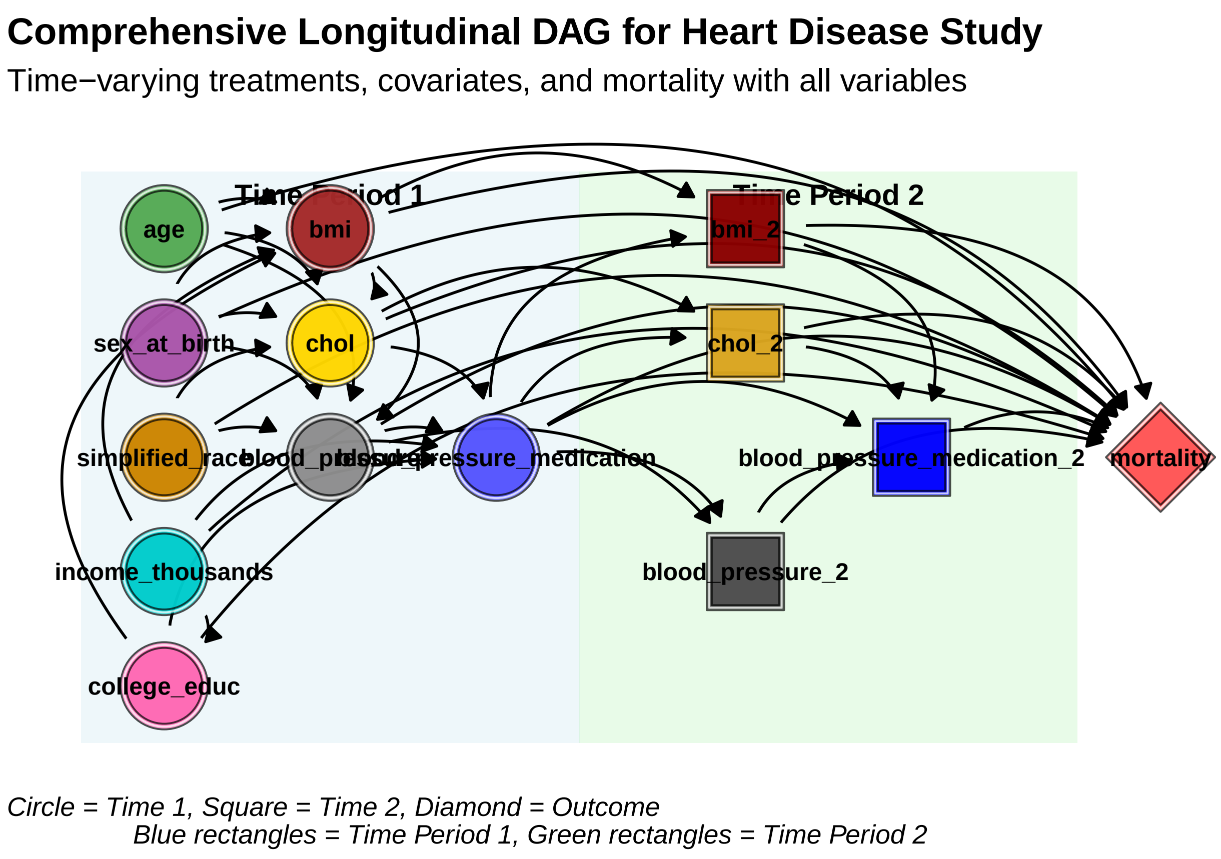
Hint: Check out slide 27 from Maya's lecture, or slides 15-17 from Dave's second slide deck in week 8 on matching.

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.

```r
# DAG for LTMLE

# Create a structured dag with all variables
structured_dag_full <- dagitty('dag {
  mortality [outcome, pos="6,0"]

  blood_pressure_medication [exposure, pos="2,0"]
  age [pos="0,2"]
  sex_at_birth [pos="0,1"]
  simplified_race [pos="0,0"]
  income_thousands [pos="0,-1"]
  college_educ [pos="0,-2"]
  bmi [pos="1,2"]
  chol [pos="1,1"]
  blood_pressure [pos="1,0"]

  blood_pressure_medication_2 [exposure, pos="4.5,0"]
  bmi_2 [pos="3.5,2"]
  chol_2 [pos="3.5,1"]
  blood_pressure_2 [pos="3.5,-1"]

  age -> bmi
  age -> blood_pressure
  age -> chol
  sex_at_birth -> bmi
  sex_at_birth -> blood_pressure
  sex_at_birth -> chol
  simplified_race -> blood_pressure
  simplified_race -> chol
  income_thousands -> college_educ
  income_thousands -> bmi
  college_educ -> bmi

  bmi -> blood_pressure
  bmi -> chol
  blood_pressure -> blood_pressure_medication
  chol -> blood_pressure_medication

  income_thousands -> blood_pressure_medication
  college_educ -> blood_pressure_medication

  blood_pressure_medication -> bmi_2
  blood_pressure_medication -> chol_2
  blood_pressure_medication -> blood_pressure_2

  bmi -> bmi_2
  chol -> chol_2
  blood_pressure -> blood_pressure_2
  blood_pressure_medication -> blood_pressure_medication_2

  bmi_2 -> blood_pressure_medication_2
  chol_2 -> blood_pressure_medication_2
  blood_pressure_2 -> blood_pressure_medication_2

  age -> mortality
  bmi -> mortality
  chol -> mortality
  blood_pressure -> mortality
  blood_pressure_medication -> mortality
  simplified_race -> mortality
  sex_at_birth -> mortality
  income_thousands -> mortality
  college_educ -> mortality

  bmi_2 -> mortality
  chol_2 -> mortality
  blood_pressure_2 -> mortality
  blood_pressure_medication_2 -> mortality
}')

# Convert to tidy format
structured_tidy_full <- tidy_dagitty(structured_dag_full)

# Prepare the time period overlay
time1_box <- data.frame(
  x = c(-0.5, 2.5, 2.5, -0.5, -0.5),
  y = c(-2.5, -2.5, 2.5, 2.5, -2.5)
)

time2_box <- data.frame(
  x = c(2.5, 5.5, 5.5, 2.5, 2.5),
  y = c(-2.5, -2.5, 2.5, 2.5, -2.5)
)

# Define time period variable sets
time1_vars <- c("age", "sex_at_birth", "simplified_race", "income_thousands",
                "college_educ", "bmi", "chol", "blood_pressure", "blood_pressure_medication")
time2_vars <- c("bmi_2", "chol_2", "blood_pressure_2", "blood_pressure_medication_2")
outcome_var <- "mortality"

# Create the updated full plot with time period boundaries
final_dag_plot_full <- ggplot() +
  # Add time period backgrounds
  geom_polygon(data = time1_box, aes(x = x, y = y), fill = "lightblue", alpha = 0.2) +
  geom_polygon(data = time2_box, aes(x = x, y = y), fill = "lightgreen", alpha = 0.2) +

  # Add time period labels
  annotate("text", x = 1, y = 2.3, label = "Time Period 1", fontface = "bold", size = 4) +
  annotate("text", x = 4, y = 2.3, label = "Time Period 2", fontface = "bold", size = 4) +

  # Add DAG edges
```

```r
geom_dag_edges_arc(data = structured_tidy_full$data,
                   aes(x = x, y = y, xend = xend, yend = yend),
                   edge_width = 0.5,
                   arrow = grid::arrow(length = grid::unit(6, "pt"), type = "closed")) +

# Time 1 nodes
geom_dag_node(data = subset(structured_tidy_full$data,
                            name %in% time1_vars) %>%
                distinct(name, x, y),
              aes(x = x, y = y, fill = name),
              size = 15, alpha = 0.7, shape = 21) +

# Time 2 nodes
geom_dag_node(data = subset(structured_tidy_full$data,
                            name %in% time2_vars) %>%
                distinct(name, x, y),
              aes(x = x, y = y, fill = name),
              size = 15, alpha = 0.7, shape = 22) +

# Outcome node
geom_dag_node(data = subset(structured_tidy_full$data, name == outcome_var) %>%
                distinct(name, x, y),
              aes(x = x, y = y, fill = name),
              size = 15, alpha = 0.7, shape = 23) +

# Add text for nodes
geom_dag_text(data = structured_tidy_full$data %>%
                distinct(name, x, y),
              aes(x = x, y = y, label = name),
              size = 3, color = "black", fontface = "bold") +

# Set node colors
scale_fill_manual(values = c(
  "mortality" = "#FF5555",
  "blood_pressure_medication" = "#5555FF",
  "blood_pressure_medication_2" = "#0000FF",
  "age" = "#55AA55",
  "sex_at_birth" = "#AA55AA",
  "simplified_race" = "#CD8500",
  "income_thousands" = "#00CDCD",
  "college_educ" = "#FF69B4",
  "bmi" = "#A52A2A",
  "bmi_2" = "#8B0000",
  "chol" = "#FFD700",
  "chol_2" = "#DAA520",
  "blood_pressure" = "#8E8E8E",
  "blood_pressure_2" = "#4D4D4D"
)) +

# Apply DAG theme
theme_dag() +

# Add titles
```

```
  labs(title = "Comprehensive Longitudinal DAG for Heart Disease Study",
       subtitle = "Time-varying treatments, covariates, and mortality with all variables",
       caption = "Circle = Time 1, Square = Time 2, Diamond = Outcome
                 Blue rectangles = Time Period 1, Green rectangles = Time Period 2") +

  # Theme adjustments
  theme(
    legend.position = "none",
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 12),
    plot.caption = element_text(face = "italic", hjust = 0)
  )

# Print the updated full DAG
print(final_dag_plot_full)
```


Comprehensive Longitudinal DAG for Heart Disease Study
Time−varying treatments, covariates, and mortality with all variables
Time Period 1
Time Period 2
age
bmi
bmi_2
sex_at_birth
chol
chol_2
simplified_race
blood_pressure
blood_pressure_medication
blood_pressure_medication_2
mortality
income_thousands
blood_pressure_2
college_educ
Circle = Time 1, Square = Time 2, Diamond = Outcome
Blue rectangles = Time Period 1, Green rectangles = Time Period 2

```
# Recalculate adjustment sets
adjustment_time1_full <- adjustmentSets(
  structured_dag_full,
  exposure = "blood_pressure_medication",
  outcome = c("mortality", "bmi_2", "chol_2", "blood_pressure_2", "blood_pressure_medication_2")
)

adjustment_time2_full <- adjustmentSets(
  structured_dag_full,
  exposure = "blood_pressure_medication_2",
  outcome = "mortality"
)

print("Adjustment sets for time 1 treatment effect (full model):")
```

```
## [1] "Adjustment sets for time 1 treatment effect (full model):"
```

```
print(adjustment_time1_full)
```

```
## { blood_pressure, chol, college_educ, income_thousands }
```

```
print("Adjustment sets for time 2 treatment effect (full model):")
```

```
## [1] "Adjustment sets for time 2 treatment effect (full model):"
```

```
print(adjustment_time2_full)
```

```
## { blood_pressure_2, blood_pressure_medication, bmi_2, chol_2 }
```

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 TMLE (no time-dependent confounding control)

# Joint treatment (e.g., ever treated over two waves)
heart_disease$ever_treated <- ifelse(
  heart_disease$blood_pressure_medication == 1 | heart_disease$blood_pressure_medication_2 == 1, 1, 0
)

# Naive TMLE model
naive_tmle <- tmle(
  Y = heart_disease$mortality,
  A = heart_disease$ever_treated,
  W = heart_disease[, c("age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands")],
  family = "binomial",
  Q.SL.library = SL.library,
  g.SL.library = SL.library
)
summary(naive_tmle)
```

```
##  Initial estimation of Q
##   Procedure: cv-SuperLearner, ensemble
##   Model:
##       Y ~  SL.mean_All + SL.glmnet_All + SL.randomForest_All + SL.xgboost_All + SL.bayesglm_All
##
```

```
##   Coefficients:
##       SL.mean_All    0
##     SL.glmnet_All    0.9518796
##   SL.randomForest_All    0.002617196
##    SL.xgboost_All    0.04550322
##   SL.bayesglm_All    0
##
##   Cross-validated pseudo R squared :  0.0188
##
##  Estimation of g (treatment mechanism)
##   Procedure: SuperLearner, ensemble
##   Model:
##       A ~  SL.mean_All + SL.glmnet_All + SL.randomForest_All + SL.xgboost_All + SL.bayesglm_All
##
##   Coefficients:
##       SL.mean_All    0
##     SL.glmnet_All    0.8039196
##   SL.randomForest_All    0.01886756
##    SL.xgboost_All    0.1772129
##   SL.bayesglm_All    0
##
##  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/ATC: (0.0054, 0.9946)
##
##  Marginal Mean under Treatment (EY1)
##    Parameter Estimate:  0.38309
##    Estimated Variance:  7.8797e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.36569, 0.40049)
##
##  Marginal Mean under Comparator (EY0)
##    Parameter Estimate:  0.56614
##    Estimated Variance:  3.2936e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.55489, 0.57739)
##
##  Additive Effect
##    Parameter Estimate:  -0.18305
##    Estimated Variance:  0.0001117
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.20377, -0.16234)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.18174
##    Estimated Variance:  0.00011777
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.20301, -0.16047)
```

```
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.18405
##    Estimated Variance:  0.00010846
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.20446, -0.16364)
##
##  Relative Risk
##    Parameter  Estimate:  0.67667
##    Variance(log scale):  0.00063953
##                p-value:  <2e-16
##      95% Conf Interval:  (0.64395, 0.71105)
##
##  Odds Ratio
##     Parameter  Estimate:  0.47589
##     Variance(log scale):  0.0019562
##                 p-value:  <2e-16
##       95% Conf Interval:  (0.43637, 0.51898)
```

```r
#LTMLE

library(ltmle)

# Prepare dataset in wide format
ltmle_data <- heart_disease[, c(
  "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",  # Baseline covariates
  "bmi", "blood_pressure", "chol",                   # Time 1 covariates
  "blood_pressure_medication",                      # Time 1 treatment
  "bmi_2", "blood_pressure_2", "chol_2",            # Time 2 covariates
  "blood_pressure_medication_2",                    # Time 2 treatment
  "mortality"                                       # Outcome
)]

# Specify node types
Anodes <- c("blood_pressure_medication", "blood_pressure_medication_2")
Lnodes <- c("bmi", "blood_pressure", "chol", "bmi_2", "blood_pressure_2", "chol_2")
Ynodes <- "mortality"

abar <- list(
  control = c(0, 0),   # Never treated at either time
  treated = c(1, 1)    # Treated at both time points
)


# Fit LTMLE model
ltmle_fit <- ltmle(
  data = ltmle_data,
  Anodes = Anodes,
  Lnodes = Lnodes,
  Ynodes = Ynodes,
  survivalOutcome = FALSE,
  SL.library = c("SL.mean", "SL.glm", "SL.bayesglm"),
  abar = abar
)
```

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

## formula for bmi_2:

## Q.kplus1 ~ age + sex_at_birth + simplified_race + college_educ +     income_thousands + bmi + blood_

## formula for mortality:

## Q.kplus1 ~ age + sex_at_birth + simplified_race + college_educ +     income_thousands + bmi + blood_

##

## gform not specified, using defaults:

## formula for blood_pressure_medication:

## blood_pressure_medication ~ age + sex_at_birth + simplified_race +     college_educ + income_thousan

## formula for blood_pressure_medication_2:

## blood_pressure_medication_2 ~ age + sex_at_birth + simplified_race +     college_educ + income_thous

##

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

```
summary(ltmle_fit)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = ltmle_data, Anodes = Anodes, Lnodes = Lnodes, Ynodes = Ynodes,
##     survivalOutcome = FALSE, abar = abar, SL.library = c("SL.mean",
##         "SL.glm", "SL.bayesglm"))
##
## Treatment Estimate:
##    Parameter Estimate:  0.56775
##     Estimated Std Err:  0.0058406
##               p-value:  <2e-16
##     95% Conf Interval: (0.5563, 0.57919)
##
## Control Estimate:
##    Parameter Estimate:  0.18799
##     Estimated Std Err:  0.033739
##               p-value:  2.5199e-08
##     95% Conf Interval: (0.12186, 0.25412)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.37976
##     Estimated Std Err:  0.034241
##               p-value:  <2e-16
##     95% Conf Interval: (0.31265, 0.44687)
##
## Relative Risk:
##    Parameter Estimate:  3.0201
##   Est Std Err log(RR):  0.17977
##              p-value:  7.8229e-10
##     95% Conf Interval: (2.1233, 4.2958)
##
## Odds Ratio:
##    Parameter Estimate:  5.6734
##   Est Std Err log(OR):  0.2223
##              p-value:  5.7938e-15
##     95% Conf Interval: (3.6696, 8.7714)
```

Above I first tried to use my fully defined SL library for the LTMLE model as I defined the library for my naive model and TMLE above, but after leaving the code running for 7 hours, I stopped it, removed the more computationally intensive/slow learners: SL.glmnet, SL.randomForest, and SL.xgboost, and I just included "SL.mean", "SL.glm", "SL.bayesglm". After removing these learners, I see that LTMLE model shows a positive treatment effect (0.38) while the naive model shows a negative treatment effect (Additive Effect Parameter Estimate: -0.18309). The fact that adjusting for time-dependent confounding reversed the effect suggests that there was substantial time-dependent confounding. There was also a big change in the magnitude of the effect (odds ratio incerases from 0.48, harmful, to 5.67, beneficial).

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'd be especially worried about health measures or biomarkers that both respond to treatment and influence subsequent treatment decisions, because these create feedback loops that bias standard analyses. Blood pressure is a good example of this because it's affected by medication at time 1, then influences medication decisions at time 2, while also independently affecting mortality risk. This creates complex pathways that standard methods cannot get at (conventional approaches might introduce collider bias by adjusting for variables on the causal pathway). A running variable like age progresses independently of treatment. While age affects treatment decisions and outcomes, treatment doesn't affect aging itself, making it different from true time-varying confounders like blood pressure, BMI, or cholesterol. I think the most problematic time-dependent confounders are those that serve as both outcomes of previous treatments and determinants of future treatments, particularly when they're strong predictors of the final outcome.