Project 8 Nereida Heller

```
#1
# Add to this package list for additional SL algorithms
pacman::p_load(
  tidyverse,
  ggthemes,
  ltmle,
  tmle,
  SuperLearner,
  tidymodels,
  caret,
  dagitty,
  ggdag,
  here)

#install.packages("xgboost")

library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(readr)
library(dplyr)
library(ggplot2)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(ggplot2)

# install.packages('tinytex')
# tinytex::install_tinytex()

#Keeping original code in case you need to run it, Kasey
#heart_disease <- read_csv(here('heart_disease_tmle.csv'))
heart_disease <- read_csv(here("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.
```

```r
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>
```

```r
glimpse(heart_disease)
```

```
## Rows: 10,000
## Columns: 14
## $ age                         <dbl> 32.92951, 53.92344, 65.33631, 16.82682, 56~
## $ sex_at_birth                <dbl> 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, ~
## $ simplified_race             <dbl> 1, 1, 3, 1, 1, 1, 1, 1, 3, 3, 3, 2, 1, 1, ~
## $ college_educ                <dbl> 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 2, 2, 2, ~
## $ income_thousands            <dbl> 91.32660, 38.76890, 35.48435, 93.77726, 85~
## $ bmi                         <dbl> 27.09986, 27.61615, 27.52499, 24.88569, 22~
## $ blood_pressure              <dbl> 146.9862, 125.7578, 131.6789, 131.2926, 13~
## $ chol                        <dbl> 211.4630, 187.8418, 197.0794, 229.6352, 20~
## $ blood_pressure_medication   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ~
## $ bmi_2                       <dbl> 27.08979, 27.61077, 27.52064, 24.89534, 22~
## $ blood_pressure_2            <dbl> 146.9800, 125.7505, 131.6626, 131.2765, 13~
## $ chol_2                      <dbl> 211.5343, 187.7360, 196.9674, 229.6558, 20~
## $ blood_pressure_medication_2 <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ~
## $ mortality                   <dbl> 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, ~
```

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

```
#2
# Subsetting to just necessary variables
heart_disease_SL <- heart_disease %>% select(-ends_with("_2"))
#glimpse(heart_disease_SL)

# Fit SuperLearner Model

## sl lib
#listWrappers()
# set seed
set.seed(987)

## Train/Test split

# initial split
# ----------
heart_split_SL <-
  initial_split(heart_disease_SL, prop = 3/4) # create initial split (tidymodels)

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

# y_train
y_train <-
  train %>%
  pull(mortality)

# x_train
x_train <-
  train %>%
  select(-mortality)

# Testing
```

```r
# ----------
test <-
  testing(heart_split_SL)

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

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

```r
# multiple models
## Train SuperLearner
# ----------
#3
sl = SuperLearner(Y = y_train,
                  X = x_train,
                  family = binomial(),
                  SL.library = c('SL.mean',     # Baseline
                                 'SL.glm',      # Logistic regression
                                 'SL.glmnet',   # Lasso and ridge
                                 'SL.ranger',   # Random forest
                                 'SL.xgboost')) #Because I remember this from earlier in the semester
```

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

```r
#4
## Risk and Coefficient of each model
sl
```

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = c("SL.mean",
##     "SL.glm", "SL.glmnet", "SL.ranger", "SL.xgboost"))
##
##
##                      Risk       Coef
## SL.mean_All    0.2497082 0.00000000
## SL.glm_All     0.2353773 0.00000000
## SL.glmnet_All  0.2352033 0.36620934
## SL.ranger_All  0.2310625 0.58237631
## SL.xgboost_All 0.2453601 0.05141435
```

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

```
## SL.ranger_All
##     0.2310625
```

```r
#5
## Confusion Matrix

# 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)
glimpse(preds)
```

```
## List of 2
##  $ pred           : num [1:2500, 1] 0.627 0.568 0.672 0.586 0.655 ...
##  $ library.predict: num [1:2500, 1:5] 0 0 0 0 0 0 0 0 0 0 ...
```

```r
# 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 --> r names them ...1 and ...2, we're renaming them s
         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`
```

```r
# confusion matrix
# table(Predicted = validation$pred, Actual = validation$obs)
# confusionMatrix(factor(validation$pred), factor(validation$obs))
print(caret::confusionMatrix(as.factor(validation$pred),
                     as.factor(validation$obs)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  410  236
##          1  808 1046
##
##                Accuracy : 0.5824
##                  95% CI : (0.5628, 0.6018)
##     No Information Rate : 0.5128
##     P-Value [Acc > NIR] : 1.684e-12
##
##                   Kappa : 0.1543
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3366
##             Specificity : 0.8159
##          Pos Pred Value : 0.6347
##          Neg Pred Value : 0.5642
##              Prevalence : 0.4872
##          Detection Rate : 0.1640
##    Detection Prevalence : 0.2584
##       Balanced Accuracy : 0.5763
##
##        'Positive' Class : 0
##
```

```
#percentages
cm <- caret::confusionMatrix(as.factor(validation$pred),
                      as.factor(validation$obs))
cm_table <- cm$table
cm_percent <- cm_table / sum(cm_table) * 100
cm_percent_rounded <- round(cm_percent, 2)
print(cm_percent_rounded)
```

```
##           Reference
## Prediction     0     1
##          0 16.40  9.44
##          1 32.32 41.84
```

```
#code with base R
# cm_table <- table(Predicted = validation$pred, Actual = validation$obs)
#
# cm_table <- cm$table
# cm_percent <- cm$table / sum(cm$table) * 100
# cm_percent_rounded <- round(cm_percent, 2)
# print(cm_percent_rounded)

# heart_split_SL
#
# print(410+236+808+1046)
```

The random forest trainer "ranger" did the best. The model didn't perform all that well in the confusion matrix though. Actual Pred TN/FN Pred FP/TP

```
      Reference
```

Prediction 0 1 0 410 236 1 808 1046

Accuracy: 0.5824 = 58.2% Recall: TP/(TP+FN) = 1046/(1046+236) = 81.6% Precision: TP/(TP+FP) = 1046/(1046+808) = 56.4%

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)?

Blending multiple algorithms will be more robust bc you end up taking advantage of the strengths of various algorithms, reducing the chance of overfitting. Superlearner also assign weights to the models, which helps mitigate the weaknesses of any one model by balancing their contributions. All of these advantages just mean that SuperLearner will outperform any single model in terms of predictive accuracy and generalization.

Targeted Maximum Likelihood Estimation

Causal Diagram

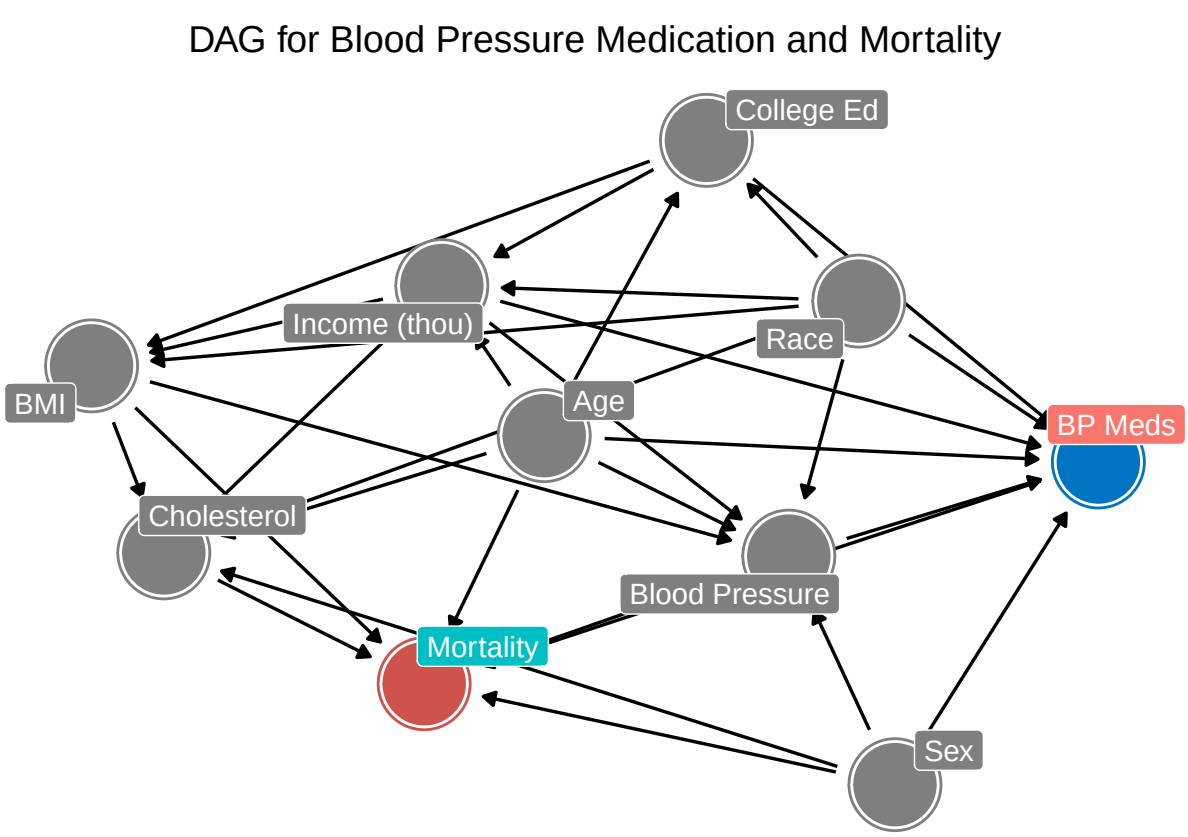
TMLE requires estimating two models:

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.

```
#6
# DAG for TMLE
blood_pressure_dag <- dagify(
  # Outcome
  mortality ~ blood_pressure_medication + blood_pressure + chol + bmi + age + sex_at_birth,

  # Treatment assignment
  blood_pressure_medication ~ blood_pressure + age + sex_at_birth + simplified_race + college_educ + in

  # Other relationships
  blood_pressure ~ age + bmi + simplified_race + sex_at_birth + income_thousands,
  chol ~ age + bmi + simplified_race + sex_at_birth + income_thousands,
  bmi ~ simplified_race + income_thousands + college_educ,
  income_thousands ~ college_educ + simplified_race + age,
  college_educ ~ simplified_race + age,

  # Exposure and outcome
  exposure = "blood_pressure_medication",
  outcome = "mortality",

  # Variable labels
  labels = c(
    "mortality" = "Mortality",
    "blood_pressure_medication" = "BP Meds",
    "blood_pressure" = "Blood Pressure",
    "chol" = "Cholesterol",
    "bmi" = "BMI",
    "age" = "Age",
    "sex_at_birth" = "Sex",
    "simplified_race" = "Race",
```

```
    "college_educ" = "College Ed",
    "income_thousands" = "Income (thou)"
  )
)

# Plot
ggdag_status(blood_pressure_dag,
             use_labels = "label",
             text = FALSE,
             stylized = TRUE) +
  theme_dag() +
  scale_color_manual(values = c("exposure" = "#0073C2",
                                "outcome" = "#CD534C",
                                "latent" = "#grey80")) +
  guides(color = "none") +  # Remove the legend
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("DAG for Blood Pressure Medication and Mortality")
```




```
#7
# # Identify adjustment sets
# adjustmentSets <- ggdag_adjustment_set(blood_pressure_dag)
# print(adjustmentSets)
#
# # Alternative view showing minimal adjustment set paths
# ggdag_paths(blood_pressure_dag) +
```

```
#   theme_dag() +
#   ggtitle("Causal Pathways in Blood Pressure Medication 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

```
#8
#troubleshooting
str(heart_disease)
```

```
## spc_tbl_ [10,000 x 14] (S3: spec_tbl_df/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 ...
##  $ bmi_2                     : num [1:10000] 27.1 27.6 27.5 24.9 22.8 ...
##  $ blood_pressure_2          : num [1:10000] 147 126 132 131 131 ...
##  $ chol_2                    : num [1:10000] 212 188 197 230 208 ...
##  $ blood_pressure_medication_2: num [1:10000] 0 0 0 0 0 0 1 0 0 0 ...
##  $ mortality                 : num [1:10000] 1 0 1 0 0 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex_at_birth = col_double(),
##   ..   simplified_race = col_double(),
##   ..   college_educ = col_double(),
##   ..   income_thousands = col_double(),
##   ..   bmi = col_double(),
##   ..   blood_pressure = col_double(),
##   ..   chol = col_double(),
##   ..   blood_pressure_medication = col_double(),
##   ..   bmi_2 = col_double(),
##   ..   blood_pressure_2 = col_double(),
##   ..   chol_2 = col_double(),
##   ..   blood_pressure_medication_2 = col_double(),
##   ..   mortality = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
heart_disease <- heart_disease %>%
  mutate(
    # Convert outcome to binary (0/1)
    mortality = as.numeric(as.factor(mortality)) - 1,
    # Convert treatment to binary
    blood_pressure_medication = as.numeric(as.factor(blood_pressure_medication)) - 1,
    # Convert categorical variables to factors
    sex_at_birth = as.factor(sex_at_birth),
    simplified_race = as.factor(simplified_race),
    college_educ = as.factor(college_educ)
  )

str(heart_disease)
```

```
## tibble [10,000 x 14] (S3: tbl_df/tbl/data.frame)
##  $ age                          : num [1:10000] 32.9 53.9 65.3 16.8 56.1 ...
##  $ sex_at_birth                 : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 1 ...
##  $ simplified_race              : Factor w/ 5 levels "1","2","3","4",..: 1 1 3 1 1 1 1 1 3 3 ...
##  $ college_educ                 : Factor w/ 2 levels "1","2": 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 ...
##  $ bmi_2                        : num [1:10000] 27.1 27.6 27.5 24.9 22.8 ...
##  $ blood_pressure_2             : num [1:10000] 147 126 132 131 131 ...
##  $ chol_2                       : num [1:10000] 212 188 197 230 208 ...
##  $ blood_pressure_medication_2: num [1:10000] 0 0 0 0 0 0 1 0 0 0 ...
##  $ mortality                    : num [1:10000] 1 0 1 0 0 1 1 1 1 1 ...
```

```r
#9
#library
SL.library <- c(
                'SL.mean',     # Baseline
                'SL.glm',      # Logistic regression
                'SL.glmnet',   # Lasso and ridge
                'SL.ranger',   # Random forest
                'SL.xgboost')

#vars
# Confounders
W <- c("blood_pressure", "age", "sex_at_birth", "simplified_race",
       "college_educ", "income_thousands", "bmi", "chol")

Y <- heart_disease$mortality                # Outcome
A <- heart_disease$blood_pressure_medication # Treatment
W_data <- heart_disease[, W]                # Confounders

#Fit TMLE model
tmle_result <- tmle(
  Y = Y,                      # Outcome: mortality
  A = A,                      # Treatment: blood pressure medication
  W = W_data,                 # Confounders from DAG
```

```
  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
  verbose = TRUE              # Print detailed output
)
```

```
##  Estimating initial regression of Y on A and W
##   using SuperLearner
##  Estimating treatment mechanism
##  Estimating missingness mechanism
##  Estimating treatment mechanism - ATT
##  Estimating treatment mechanism - ATC
```

```
#10
#print results
print(tmle_result)
```

```
##  Marginal mean under treatment (EY1)
##    Parameter Estimate:  0.21388
##    Estimated Variance:  3.2117e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.20277, 0.22498)
##
##  Marginal mean under comparator (EY0)
##    Parameter Estimate:  0.56639
##    Estimated Variance:  2.7927e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (0.55603, 0.57675)
##
##  Additive Effect
##    Parameter Estimate:  -0.35252
##    Estimated Variance:  5.994e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.36769, -0.33734)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.74822
##    Estimated Variance:  2.2293e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.75747, -0.73897)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.56518
##    Estimated Variance:  2.7441e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.57545, -0.55491)
##
##  Relative Risk
##    Parameter  Estimate: 0.37761
##    Variance(log scale): 0.00078832
##                p-value: <2e-16
##      95% Conf Interval: (0.35739, 0.39897)
##
```

```
##  Odds Ratio
##    Parameter  Estimate: 0.20828
##    Variance(log scale): 0.0015966
##                 p-value: <2e-16
##      95% Conf Interval: (0.19259, 0.22525)
```

```r
#extract
ate <- tmle_result$estimates$ATE
ate_ci_lower <- ate$CI[1]
ate_ci_upper <- ate$CI[2]
ate_pvalue <- ate$pvalue

#quick check
print(ate)
```

```
## $psi
## [1] -0.3525159
##
## $var.psi
## [1] 5.994008e-05
##
## $CI
## [1] -0.3676901 -0.3373417
##
## $pvalue
## [1] 0
##
## $bs.var
## [1] NA
##
## $bs.CI.twosided
## [1] NA NA
##
## $bs.CI.onesided.lower
## [1] -Inf   NA
##
## $bs.CI.onesided.upper
## [1]  NA Inf
```

```r
print(ate_ci_lower)
```

```
## [1] -0.3676901
```

```r
print(ate_ci_upper)
```

```
## [1] -0.3373417
```

```r
print(ate_pvalue)
```

```
## [1] 0
```

```r
#store results in df
results_df <- data.frame(
  Estimator = c("Initial", "TMLE"),
  ATE = c(tmle_result$estimates$ATE$initial, tmle_result$estimates$ATE$psi),
  SE = c(NA, tmle_result$estimates$ATE$se),
  p_value = c(NA, tmle_result$estimates$ATE$pvalue),
  CI_lower = c(NA, tmle_result$estimates$ATE$CI[1]),
  CI_upper = c(NA, tmle_result$estimates$ATE$CI[2])
)

#print
cat("\n\n--- TMLE Results Summary ---\n")
```

```
##
##
## --- TMLE Results Summary ---
```

```r
cat("Average Treatment Effect (ATE): ", round(ate$psi, 4), "\n")
```

```
## Average Treatment Effect (ATE):  -0.3525
```

```r
cat("95% Confidence Interval: [", round(ate_ci_lower, 4), ", ", round(ate_ci_upper, 4), "]\n")
```

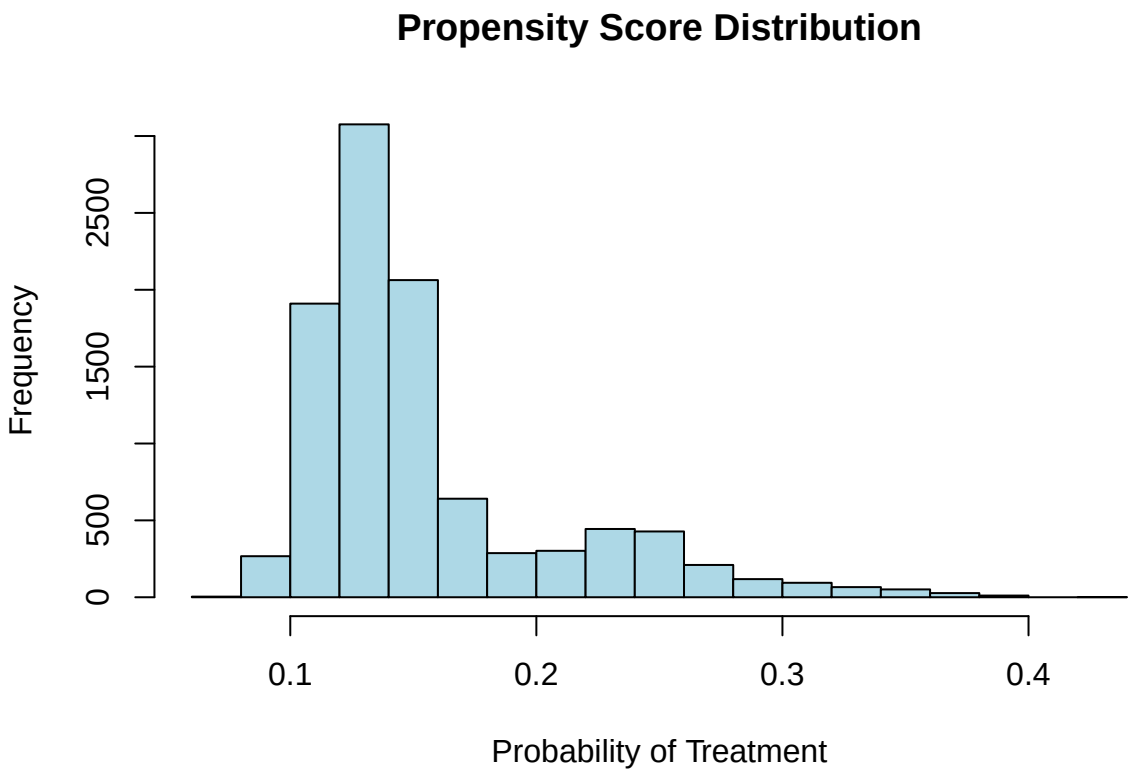
```
## 95% Confidence Interval: [ -0.3677 ,  -0.3373 ]
```

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

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

So this is telling us that taking blood pressure medication decreases mortality risk by 35.25 % on average.

```r
#11
#trying to understand treatment distribution, it's pretty rare to get treatment, not actually sure afte

#propensity score dist
prop_scores <- tmle_result$g$g1W
hist(prop_scores, main="Propensity Score Distribution",
     xlab="Probability of Treatment", col="lightblue", breaks=20)
```




```
ps_min <- min(prop_scores)
ps_max <- max(prop_scores)
ps_imbalance <- data.frame(
  Group = c("Treated", "Control"),
  Count = c(sum(A == 1), sum(A == 0)),
  Mean_PS = c(mean(prop_scores[A == 1]), mean(prop_scores[A == 0]))
)

cat("\n--- Propensity Score Diagnostics ---\n")
```

```
##
## --- Propensity Score Diagnostics ---
```

```
cat("Minimum propensity score: ", round(ps_min, 4), "\n")
```

```
## Minimum propensity score:  0.0752
```

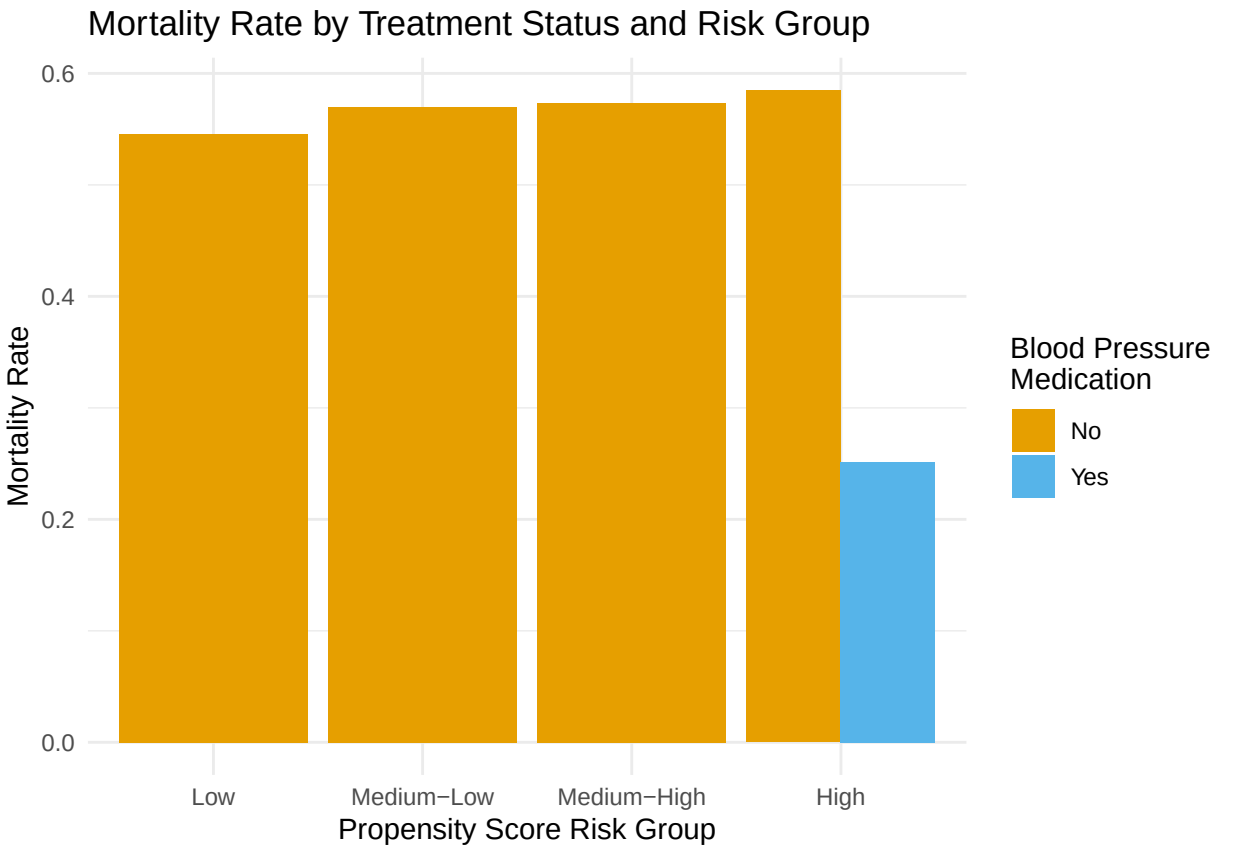
```
cat("Maximum propensity score: ", round(ps_max, 4), "\n")
```

```
## Maximum propensity score:  0.4242
```

```
print(ps_imbalance)
```

```
##     Group Count   Mean_PS
## 1 Treated  1551 0.2597029
## 2 Control  8449 0.1361839
```

```r
#12

# By risk group
heart_disease$prop_score <- prop_scores
heart_disease$risk_group <- cut(heart_disease$prop_score,
                                breaks = quantile(heart_disease$prop_score, probs = seq(0, 1, 0.25)),
                                labels = c("Low", "Medium-Low", "Medium-High", "High"),
                                include.lowest = TRUE)
# Mortality rates by treatment and risk group
effect_by_risk <- heart_disease %>%
  group_by(risk_group, blood_pressure_medication) %>%
  summarize(
    mortality_rate = mean(mortality),
    count = n(),
    .groups = "drop"
  )



# Treatment effect by risk group plot
ggplot(effect_by_risk, aes(x = risk_group, y = mortality_rate,
                           fill = factor(blood_pressure_medication))) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("#E69F00", "#56B4E9"),
                    name = "Blood Pressure\nMedication",
                    labels = c("No", "Yes")) +
  labs(title = "Mortality Rate by Treatment Status and Risk Group",
       x = "Propensity Score Risk Group",
       y = "Mortality Rate") +
  theme_minimal() +
  theme(legend.position = "right")
```



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 are those that combine outcome and propensity score methods. They're helpful because they still work correctly if either your outcome model or propensity score model is wrong (but not both). When your outcome model is right but propensity scores are off, the outcome model gives you consistent estimates; When the propensity model is right but outcome model is wrong, the balancing between groups still allows the analysis to be unbiased. This gives you a kind of a safety net compared to methods that might fail completely of the model is misspecified. Traditional statistics would emphasize the outcome modeling piece; earlier with the matching material we looked at how to using matching methods to balance treated and control groups without requiring a correctly specified outcome model. Double robust estimators do both.

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

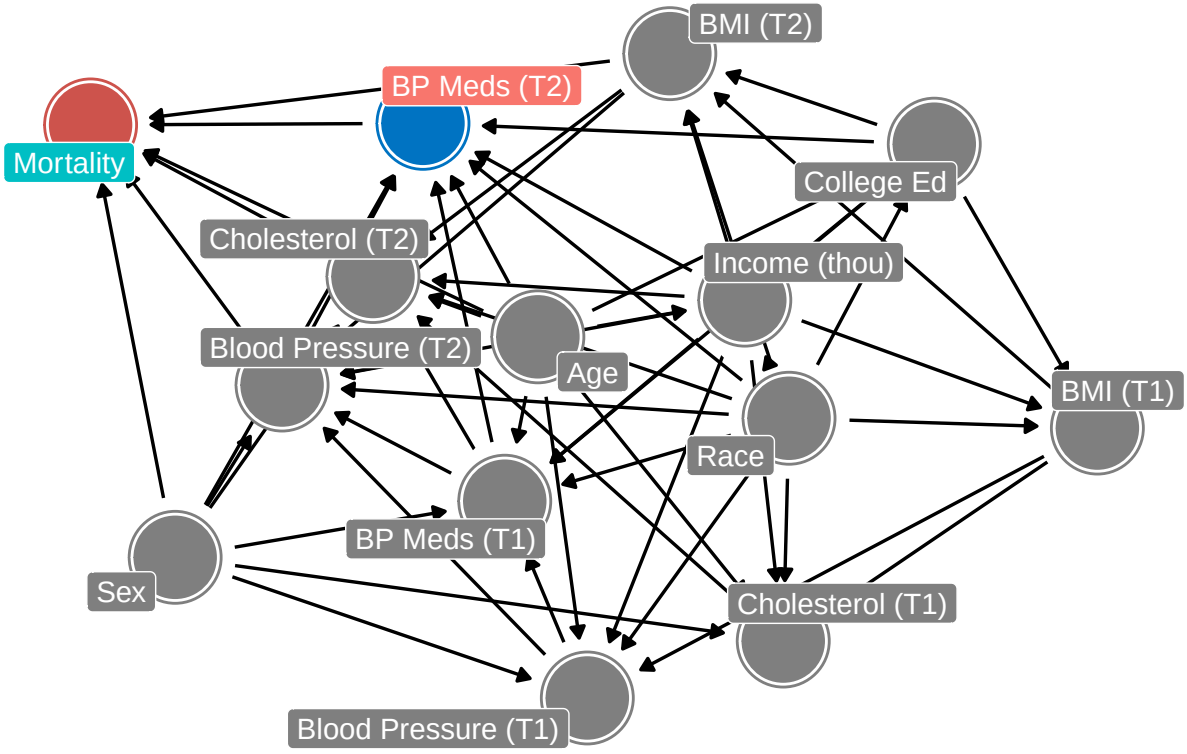
Update your causal diagram to incorporate this new information. **Note**: If your groups divides up sections and someone is working on LTMLE separately from TMLE then just draw a causal diagram even if it does not match the one you specified above.

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.

```
#13
# DAG for LTMLE

blood_pressure_dag <- dagify(
  # Outcome
  mortality ~ blood_pressure_medication_2 + blood_pressure_2 + chol_2 + bmi_2 + age + sex_at_birth,

  # Treatment assignment - Time 2
  blood_pressure_medication_2 ~ blood_pressure_2 + blood_pressure_medication + age + sex_at_birth + simp

  # Time 2 measurements influenced by time 1 measurements
  blood_pressure_2 ~ blood_pressure + blood_pressure_medication + age + bmi_2 + simplified_race + sex_at
  chol_2 ~ chol + blood_pressure_medication + age + bmi_2 + simplified_race + sex_at_birth + income_thou
  bmi_2 ~ bmi + simplified_race + income_thousands + college_educ,

  # Treatment assignment - Time 1
  blood_pressure_medication ~ blood_pressure + age + sex_at_birth + simplified_race + college_educ + in

  # Other relationships - Time 1
  blood_pressure ~ age + bmi + simplified_race + sex_at_birth + income_thousands,
  chol ~ age + bmi + simplified_race + sex_at_birth + income_thousands,
  bmi ~ simplified_race + income_thousands + college_educ,
  income_thousands ~ college_educ + simplified_race + age,
  college_educ ~ simplified_race + age,

  # Exposure and outcome
  exposure = "blood_pressure_medication_2",
  outcome = "mortality",

  # Variable labels
  labels = c(
    "mortality" = "Mortality",
    "blood_pressure_medication" = "BP Meds (T1)",
    "blood_pressure_medication_2" = "BP Meds (T2)",
    "blood_pressure" = "Blood Pressure (T1)",
    "blood_pressure_2" = "Blood Pressure (T2)",
    "chol" = "Cholesterol (T1)",
    "chol_2" = "Cholesterol (T2)",
    "bmi" = "BMI (T1)",
    "bmi_2" = "BMI (T2)",
    "age" = "Age",
    "sex_at_birth" = "Sex",
    "simplified_race" = "Race",
```

```
    "college_educ" = "College Ed",
    "income_thousands" = "Income (thou)"
  )
)

# Plot
ggdag_status(blood_pressure_dag,
             use_labels = "label",
             text = FALSE,
             stylized = TRUE) +
  theme_dag() +
  scale_color_manual(values = c("exposure" = "#0073C2",
                                "outcome" = "#CD534C",
                                "latent" = "#grey80")) +
  guides(color = "none") +  # Remove the legend
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("DAG for Blood Pressure Medication and Mortality (with Time Period 2 Vars)")
```



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: Outcomes look worse in treatment than control (52.16% vs 51.63%), but difference is not statistically significant, which is good because we don't want to think that the medication is actually harmful!

2 time-points: The results of the LTLMLE with 1,000 observations and the full SL library and two time-points shows a positive impact of treatment (31.55 percentage points) that's statistically significant. P values are nicely small, confidence intervals don't include zero, risk and odds and additive effect all point towards a real treatment effect.

Nb: I couldn't run the LTMLE with all the observations – every time I tried, R crashed. I was able to run it with 1000 observations once, but that took forever. Below, you'll see all my little experiments on how to get a result without R crashing. The only version of the LTMLE that I haven't commented out is the one with 500 observations and the full SL library.

```
#14
## Naive Model (no time-dependent confounding) estimate
SL.library <- c(
  'SL.mean',    # Baseline
  'SL.glm',     # Logistic regression
  'SL.glmnet',  # Lasso and ridge
  'SL.ranger',  # Random forest
  'SL.xgboost'  # Gradient boosting
)

#checking
print(str(heart_disease))
```

```
## tibble [10,000 x 16] (S3: tbl_df/tbl/data.frame)
##  $ age                         : num [1:10000] 32.9 53.9 65.3 16.8 56.1 ...
##  $ sex_at_birth                : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 1 ...
##  $ simplified_race             : Factor w/ 5 levels "1","2","3","4",..: 1 1 3 1 1 1 1 1 3 3 ...
##  $ college_educ                : Factor w/ 2 levels "1","2": 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 ...
##  $ bmi_2                       : num [1:10000] 27.1 27.6 27.5 24.9 22.8 ...
##  $ blood_pressure_2            : num [1:10000] 147 126 132 131 131 ...
##  $ chol_2                      : num [1:10000] 212 188 197 230 208 ...
##  $ blood_pressure_medication_2: num [1:10000] 0 0 0 0 0 0 1 0 0 0 ...
##  $ mortality                   : num [1:10000] 1 0 1 0 0 1 1 1 1 1 ...
##  $ prop_score                  : num [1:10000] 0.144 0.132 0.136 0.114 0.154 ...
##  $ risk_group                  : Factor w/ 4 levels "Low","Medium-Low",..: 3 2 2 1 3 3 1 3 1 1 ...
## NULL
```

```
print(head(heart_disease))
```

```
## # A tibble: 6 x 16
##     age sex_at_birth simplified_race college_educ income_thousands   bmi
##   <dbl> <fct>        <fct>           <fct>                    <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 10 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>,
## #   prop_score <dbl>, risk_group <fct>
```

```
# #for naive model, use only second time period data?
# W <- c("blood_pressure_2", "age", "sex_at_birth", "simplified_race",
#        "college_educ", "income_thousands", "bmi_2", "chol_2")
#
# Y <- heart_disease$mortality                          # Outcome
# A <- heart_disease$blood_pressure_medication_2        # Treatment at time 2
# W_data <- heart_disease[, W]                          # Confounders
#
#
# # Fit naive TMLE
# naive_result <- tmle::tmle(
#   Y = Y,                      # Outcome: mortality
#   A = A,                      # Treatment: blood pressure medication at time 2
#   W = W_data,                 # Confounders at time 2
#   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
#   verbose = TRUE              # Print detailed output
# )

#data
##baseline
baseline_vars <- c("age", "sex_at_birth", "simplified_race",
                   "college_educ", "income_thousands")

## Create data for naive approach (I don't know if this is necessary?)
naive_data <- heart_disease[, c(baseline_vars, "blood_pressure_medication_2", "mortality")]
colnames(naive_data)[colnames(naive_data) == "blood_pressure_medication_2"] <- "A"
colnames(naive_data)[colnames(naive_data) == "mortality"] <- "Y"

# Run naive LTMLE approach (ignoring time-dependency)
naive_ltmle_result <- ltmle(
  data = naive_data,
  Anodes = "A",      # Only final treatment
  Lnodes = NULL,     # No time-dependent confounders
  Ynodes = "Y",      # Outcome
  abar = list(1, 0), # Compare treatment vs no treatment
  SL.library = SL.library
)
```

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

## formula for Y:

## Q.kplus1 ~ age + sex_at_birth + simplified_race + college_educ +     income_thousands + A

##
```

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

## formula for A:

## A ~ age + sex_at_birth + simplified_race + college_educ + income_thousands

##

## Estimate of time to completion: 4 to 16 minutes
```

```
summary(naive_ltmle_result)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = naive_data, Anodes = "A", Lnodes = NULL, Ynodes = "Y",
##     abar = list(1, 0), SL.library = SL.library)
##
## Treatment Estimate:
##    Parameter Estimate:  0.52138
##     Estimated Std Err:  0.013607
##               p-value:  <2e-16
##     95% Conf Interval: (0.49472, 0.54805)
##
## Control Estimate:
##    Parameter Estimate:  0.51625
##     Estimated Std Err:  0.0053735
##               p-value:  <2e-16
##     95% Conf Interval: (0.50571, 0.52678)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.0051394
##     Estimated Std Err:  0.014629
##               p-value:  0.72536
##     95% Conf Interval: (-0.023534, 0.033813)
##
## Relative Risk:
##    Parameter Estimate:  1.01
##   Est Std Err log(RR):  0.028097
##               p-value:  0.72441
##     95% Conf Interval: (0.95584, 1.0671)
##
## Odds Ratio:
##    Parameter Estimate:  1.0208
##   Est Std Err log(OR):  0.058619
##               p-value:  0.72544
##     95% Conf Interval: (0.91001, 1.1451)
```

```
# # Naive -- tried this using ltmle code from class and I couldn't figure it out
# naive_result2 <- ltmle(
#   heart_disease,
#   Anodes = A,
#   Lnodes = NULL,
```

```
#   Ynodes = Y,
#   abar =
#   #Q.SL.library = SL.library,  # SuperLearner library for outcome model
#   #g.SL.library = SL.library,  # SuperLearner library for propensity score model
# )


#print("Naive TMLE Results (ignoring time-dependent confounding):")
# print(naive_result)
# print(naive_result2)
```

Below there are a few attempts at running the full LTMLE. R kept quitting whenever I run it. I'm going to try a few things: subset the data, run the same code with a smaller library, to see if that helps at all.

```
#15
## LTMLE estimate
#
# #data
# ltmle_data <- heart_disease
#
# # Nodes for ltmle:
# # L1: Baseline covariates (time 1)
# # A1: Treatment at time 1
# # L2: Time-varying covariates at time 2
# # A2: Treatment at time 2
# # Y: Outcome
#
# # Rename variables to match ltmle naming convention
# names(ltmle_data)[names(ltmle_data) == "blood_pressure"] <- "L1_bp"
# names(ltmle_data)[names(ltmle_data) == "bmi"] <- "L1_bmi"
# names(ltmle_data)[names(ltmle_data) == "chol"] <- "L1_chol"
# names(ltmle_data)[names(ltmle_data) == "blood_pressure_medication"] <- "A1"
# names(ltmle_data)[names(ltmle_data) == "blood_pressure_2"] <- "L2_bp"
# names(ltmle_data)[names(ltmle_data) == "bmi_2"] <- "L2_bmi"
# names(ltmle_data)[names(ltmle_data) == "chol_2"] <- "L2_chol"
# names(ltmle_data)[names(ltmle_data) == "blood_pressure_medication_2"] <- "A2"
# names(ltmle_data)[names(ltmle_data) == "mortality"] <- "Y"
#
# # Define node lists for ltmle based on the DAG
# Lnodes <- c("L1_bp", "L1_bmi", "L1_chol", "age", "sex_at_birth", "simplified_race",
#             "college_educ", "income_thousands", "L2_bp", "L2_bmi", "L2_chol")
# Anodes <- c("A1", "A2")
# Ynodes <- "Y"
#
# # Define treatment regimens to compare
# # 1. Always treated: Treatment at both time points (1,1)
# # 2. Never treated: No treatment at both time points (0,0)
# regimen_treated <- c(1, 1)
# regimen_control <- c(0, 0)
#
# # Verify data structure before fitting ltmle
# print("LTMLE data structure:")
# print(names(ltmle_data))
```

```
#
# # Fit ltmle model
# # We need to separate the time-varying covariates into time-specific nodes for proper time ordering
# # Based on the DAG:
# # - At time 1: blood pressure, bmi, chol are affected by age, sex, race, income
# # - These in turn affect treatment assignment
# Lnodes1 <- c("L1_bp", "L1_bmi", "L1_chol", "age", "sex_at_birth", "simplified_race",
#              "college_educ", "income_thousands")
# Lnodes2 <- c("L2_bp", "L2_bmi", "L2_chol")
#
#
# # Specify custom formulas that reflect the DAG structure
# # From the DAG:
# # - blood_pressure_medication depends on blood_pressure, age, sex_at_birth, race, education, income
# # - mortality depends on medication, blood_pressure, chol, bmi, age, sex
#
# # Create a named list of formulas for the Q models (outcome regressions)
# Qform <- c(
#   Y = "Q.kplus1 ~ A2 + L2_bp + L2_bmi + L2_chol + age + sex_at_birth",
#   # For time 2 nodes, include previous treatment and baseline
#   L2_bp = "L2_bp ~ A1 + L1_bp + age + L1_bmi + simplified_race + sex_at_birth + income_thousands",
#   L2_bmi = "L2_bmi ~ A1 + L1_bmi + simplified_race + income_thousands + college_educ",
#   L2_chol = "L2_chol ~ A1 + L1_chol + age + L1_bmi + simplified_race + sex_at_birth + income_thousand
# )
#
# # Create a named list of formulas for the g models (treatment regressions)
# gform <- c(
#   # Treatment at time 1 depends on baseline variables
#   A1 = "A1 ~ L1_bp + age + sex_at_birth + simplified_race + college_educ + income_thousands",
#   # Treatment at time 2 depends on previous treatment and updated covariates
#   A2 = "A2 ~ A1 + L2_bp + age + sex_at_birth + simplified_race + college_educ + income_thousands"
# )
#
# # Fit the ltmle model -- ERROR HERE
# ltmle_result <- ltmle(
#   data = ltmle_data,
#   Anodes = Anodes,
#   Lnodes = list(Lnodes1, Lnodes2),  # Specify time-specific L nodes
#   Ynodes = Ynodes,
#   abar = list(regimen_treated, regimen_control),  # Compare always treated vs never treated
#   SL.library = SL.library,
#   Qform = Qform,  # Use DAG-informed outcome models
#   gform = gform,  # Use DAG-informed treatment models
#   variance.method = "tmle"
# )
#
# # Print ltmle results
# print("Longitudinal TMLE Results (accounting for time-dependent confounding):")
# print(summary(ltmle_result))
#
# # Extract and compare results
# naive_estimate <- naive_result$estimates$ATE$psi
# naive_ci_lower <- naive_result$estimates$ATE$CI[1]
```

```
# naive_ci_upper <- naive_result$estimates$ATE$CI[2]
#
# ltmle_summary <- summary(ltmle_result)
# ltmle_estimate <- ltmle_summary$effect.measures$ATE$estimate
# ltmle_ci_lower <- ltmle_summary$effect.measures$ATE$CI[1]
# ltmle_ci_upper <- ltmle_summary$effect.measures$ATE$CI[2]
#
# results_df <- data.frame(
#   Model = c("Naive TMLE", "Longitudinal TMLE"),
#   Estimate = c(naive_estimate, ltmle_estimate),
#   CI_Lower = c(naive_ci_lower, ltmle_ci_lower),
#   CI_Upper = c(naive_ci_upper, ltmle_ci_upper)
# )
#
# print("Comparison of Estimates:")
# print(results_df)
#
# # Create a visualization of the results
# ggplot(results_df, aes(x = Model, y = Estimate)) +
#   geom_point(size = 3) +
#   geom_errorbar(aes(ymin = CI_Lower, ymax = CI_Upper), width = 0.2) +
#   geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
#   labs(title = "Comparison of Treatment Effect Estimates",
#        subtitle = "Effect of Blood Pressure Medication on Mortality",
#        y = "Average Treatment Effect",
#        x = "") +
#   theme_minimal() +
#   theme(axis.text.x = element_text(angle = 0, hjust = 0.5),
#         plot.title = element_text(hjust = 0.5),
#         plot.subtitle = element_text(hjust = 0.5))
#
# # Calculate the difference between estimates
# difference <- ltmle_estimate - naive_estimate
# difference_percent <- (difference / abs(naive_estimate)) * 100
#
# # Print difference
# cat("\nDifference between estimates (LTMLE - Naive):", round(difference, 4), "\n")
# cat("Percent difference:", round(difference_percent, 2), "%\n")
#
# # Interpretation helper
# if (abs(difference_percent) > 10) {
#   cat("\nThe difference between estimates is substantial (>10%), suggesting that\n")
#   cat("time-dependent confounding has a meaningful impact on the treatment effect estimate.\n")
# } else {
#   cat("\nThe difference between estimates is modest (<10%), suggesting that\n")
#   cat("time-dependent confounding may not have a major impact in this specific case.\n")
# }
#
# # Further analysis to understand the source of differences
# # Check if treatment assignment at time 2 depends on time-varying covariates
# cat("\nAnalyzing potential time-dependent confounding based on the DAG structure:\n")
#
# # Based on the DAG, check the relationship between time 1 variables and treatment at time 2
```

```
# # The DAG shows blood_pressure_medication is influenced by blood_pressure, age, sex_at_birth, etc.
# print("Relationship between time 1 variables and treatment at time 2 (based on DAG):")
# time_confounding_model <- glm(A2 ~ L1_bp + A1 + age + sex_at_birth + simplified_race +
#                               college_educ + income_thousands,
#                             family = binomial, data = ltmle_data)
# print(summary(time_confounding_model))
#
# # Check if time-varying covariates at time 2 are associated with treatment and outcome
# # Based on the DAG, mortality is influenced by blood_pressure_medication, blood_pressure, chol, bmi,
# print("Relationship between time 2 variables, treatment, and outcome (based on DAG):")
# outcome_model <- glm(Y ~ A2 + L2_bp + L2_bmi + L2_chol + age + sex_at_birth,
#                      family = binomial, data = ltmle_data)
# print(summary(outcome_model))
```

##Second attempt

```
# # Preparing data for proper LTMLE approach
# #16
# #I don't think my computer can handle this, actually. I am going to subset the data and try a smaller
#
#
# ##troubleshooting
# data_ordered <- heart_disease[, c(
#   # Baseline covariates and baseline measurements
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
#   "bmi", "blood_pressure", "chol",
#
#   # Time 1 treatment
#   "blood_pressure_medication",
#
#   # Time 2 measurements
#   "bmi_2", "blood_pressure_2", "chol_2",
#
#   # Time 2 treatment
#   "blood_pressure_medication_2",
#
#   # Outcome
#   "mortality"
# )]
#
#
# # Define SuperLearner library
# SL.library <- c(
#   'SL.mean',    # Baseline
#   'SL.glm',     # Logistic regression
#   'SL.glmnet',  # Lasso and ridge
#   'SL.ranger',  # Random forest
#   'SL.xgboost'  # Gradient boosting
# )
#
#
# # Ltmle nodes order:
# # 1. Baseline covariates (Lnodes)
```

```
# # 2. Time 1 treatment (Anodes)
# # 3. Time 2 covariates (Lnodes)
# # 4. Time 2 treatment (Anodes)
# # 5. Outcome (Ynodes)
#
#
# # Correct order?
# Lnodes <- c(
#   # Baseline covariates
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
#   "bmi", "blood_pressure", "chol",
#
#   # Time 2 measurements (these come after time 1 treatment)
#   "bmi_2", "blood_pressure_2", "chol_2"
# )
#
# Anodes <- c("blood_pressure_medication", "blood_pressure_medication_2")
# Ynodes <- "mortality"
#
# # Run full LTMLE with proper ordering
# full_ltmle_result <- ltmle(
#   data = data_ordered, # Use the ordered dataset
#   Anodes = Anodes,
#   Lnodes = Lnodes,
#   Ynodes = Ynodes,
#   abar = list(c(1,1), c(0,0)),
#   SL.library = SL.library
# )
#
# summary(full_ltmle_result)
```

```
#17
str(heart_disease)
```

```
## tibble [10,000 x 16] (S3: tbl_df/tbl/data.frame)
##  $ age                        : num [1:10000] 32.9 53.9 65.3 16.8 56.1 ...
##  $ sex_at_birth               : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 1 ...
##  $ simplified_race            : Factor w/ 5 levels "1","2","3","4",..: 1 1 3 1 1 1 1 1 3 3 ...
##  $ college_educ               : Factor w/ 2 levels "1","2": 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 ...
##  $ bmi_2                      : num [1:10000] 27.1 27.6 27.5 24.9 22.8 ...
##  $ blood_pressure_2           : num [1:10000] 147 126 132 131 131 ...
##  $ chol_2                     : num [1:10000] 212 188 197 230 208 ...
##  $ blood_pressure_medication_2: num [1:10000] 0 0 0 0 0 0 1 0 0 0 ...
##  $ mortality                  : num [1:10000] 1 0 1 0 0 1 1 1 1 1 ...
##  $ prop_score                 : num [1:10000] 0.144 0.132 0.136 0.114 0.154 ...
##  $ risk_group                 : Factor w/ 4 levels "Low","Medium-Low",..: 3 2 2 1 3 3 1 3 1 1 ...
```

```r
names(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"
## [15] "prop_score"                   "risk_group"
```

Subset data

```r
#18
## Subset data -- Nb this actually ran. Going to do a few more experiments below but this might be the



# Define SuperLearner library (same as)
SL.library <- c(
  'SL.mean',     # Baseline
  'SL.glm',      # Logistic regression
  'SL.glmnet',   # Lasso and ridge
  'SL.ranger',   # Random forest
  'SL.xgboost'   # Gradient boosting
)


# Ltmle nodes order:
# 1. Baseline covariates (Lnodes)
# 2. Time 1 treatment (Anodes)
# 3. Time 2 covariates (Lnodes)
# 4. Time 2 treatment (Anodes)
# 5. Outcome (Ynodes)

set.seed(123)

# Take a random sample of 500 observations
data_subset <- heart_disease[sample(nrow(heart_disease), 500), ]


# order
data_ordered <- data_subset[, c(
  # Baseline covariates and baseline measurements
  "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
  "bmi", "blood_pressure", "chol",

  # Time 1 treatment
  "blood_pressure_medication",

  # Time 2 measurements
```

```r
  "bmi_2", "blood_pressure_2", "chol_2",

  # Time 2 treatment
  "blood_pressure_medication_2",

  # Outcome
  "mortality"
)]

# Correct order?
Lnodes <- c(
  # Baseline covariates
  "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
  "bmi", "blood_pressure", "chol",

  # Time 2 measurements (these come after time 1 treatment)
  "bmi_2", "blood_pressure_2", "chol_2"
)

Anodes <- c("blood_pressure_medication", "blood_pressure_medication_2")
Ynodes <- "mortality"

# Run full LTMLE with proper ordering
full_ltmle_result <- ltmle(
  data = data_ordered, # Use the ordered dataset
  Anodes = Anodes,
  Lnodes = Lnodes,
  Ynodes = Ynodes,
  abar = list(c(1,1), c(0,0)),
  SL.library = SL.library
)
```

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

## formula for bmi_2:

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

## formula for mortality:

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

##

## gform not specified, using defaults:

## formula for blood_pressure_medication:

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

## formula for blood_pressure_medication_2:
```

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

##

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in h(simpleError(msg, call)) :
##   error in evaluating the argument 'x' in selecting a method for function 'drop': non-conformable ar
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
```

```
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds
## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
## Error in pred[, "1"] : subscript out of bounds

## Estimate of time to completion: 1 to 2 minutes

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
```

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed
```

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

## Error in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs,  :
##   one multinomial or binomial class has 1 or 0 observations; not allowed

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

```r
summary(full_ltmle_result)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = data_ordered, Anodes = Anodes, Lnodes = Lnodes,
##     Ynodes = Ynodes, abar = list(c(1, 1), c(0, 0)), SL.library = SL.library)
##
## Treatment Estimate:
##    Parameter Estimate:  0.21801
##     Estimated Std Err:  0.13278
##               p-value:  0.1006
##     95% Conf Interval: (0, 0.47825)
##
## Control Estimate:
##    Parameter Estimate:  0.59573
##     Estimated Std Err:  0.02619
##               p-value:  <2e-16
##     95% Conf Interval: (0.5444, 0.64706)
##
## Additive Treatment Effect:
##    Parameter Estimate:  -0.37772
##     Estimated Std Err:  0.13533
##               p-value:  0.0052544
##     95% Conf Interval: (-0.64297, -0.11247)
##
## Relative Risk:
##    Parameter Estimate:  0.36595
##   Est Std Err log(RR):  0.61062
##               p-value:  0.099709
##     95% Conf Interval: (0.11058, 1.2111)
##
## Odds Ratio:
##    Parameter Estimate:  0.18919
##   Est Std Err log(OR):  0.78638
##               p-value:  0.034235
##     95% Conf Interval: (0.040507, 0.88363)
```

Smaller library

```r
#19
# # Smaller library -- this won't run, R crashes
#
# ##troubleshooting
# data_ordered <- heart_disease[, c(
#   # Baseline covariates and baseline measurements
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
#   "bmi", "blood_pressure", "chol",
#
#   # Time 1 treatment
#   "blood_pressure_medication",
#
```

```
#   # Time 2 measurements
#   "bmi_2", "blood_pressure_2", "chol_2",
#
#   # Time 2 treatment
#   "blood_pressure_medication_2",
#
#   # Outcome
#   "mortality"
# )]
#
#
# # Define SuperLearner library -- taking out a few to see what happens
# SL.library <- c(
#   'SL.mean',    # Baseline
#   #'SL.glm',     # Logistic regression
#   'SL.glmnet',  # Lasso and ridge
#   'SL.ranger'  # Random forest
#   #'SL.xgboost'  # Gradient boosting
# )
#
#
# # Ltmle nodes order:
# # 1. Baseline covariates (Lnodes)
# # 2. Time 1 treatment (Anodes)
# # 3. Time 2 covariates (Lnodes)
# # 4. Time 2 treatment (Anodes)
# # 5. Outcome (Ynodes)
#
#
# # Correct order?
# Lnodes <- c(
#   # Baseline covariates
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
#   "bmi", "blood_pressure", "chol",
#
#   # Time 2 measurements (these come after time 1 treatment)
#   "bmi_2", "blood_pressure_2", "chol_2"
# )
#
# Anodes <- c("blood_pressure_medication", "blood_pressure_medication_2")
# Ynodes <- "mortality"
#
# # Run full LTMLE with proper ordering
# full_ltmle_result <- ltmle(
#   data = data_ordered, # Use the ordered dataset
#   Anodes = Anodes,
#   Lnodes = Lnodes,
#   Ynodes = Ynodes,
#   abar = list(c(1,1), c(0,0)),
#   SL.library = SL.library
# )
#
# summary(full_ltmle_result)
```

Subset of data and smaller library

```
#
# #20
# ## Subset data + small library
#
#
#
# # Define SuperLearner library (smaller)
# SL.library <- c(
#   'SL.mean',    # Baseline
#   #'SL.glm',     # Logistic regression
#   'SL.glmnet',  # Lasso and ridge
#   'SL.ranger'  # Random forest
#   #'SL.xgboost'  # Gradient boosting
# )
#
# # Ltmle nodes order:
# # 1. Baseline covariates (Lnodes)
# # 2. Time 1 treatment (Anodes)
# # 3. Time 2 covariates (Lnodes)
# # 4. Time 2 treatment (Anodes)
# # 5. Outcome (Ynodes)
#
# set.seed(123)
#
# # Take a random sample of observations -- trying 1000 this time
# data_subset <- heart_disease[sample(nrow(heart_disease), 1000), ]
#
#
# # order
# data_ordered <- data_subset[, c(
#   # Baseline covariates and baseline measurements
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
#   "bmi", "blood_pressure", "chol",
#
#   # Time 1 treatment
#   "blood_pressure_medication",
#
#   # Time 2 measurements
#   "bmi_2", "blood_pressure_2", "chol_2",
#
#   # Time 2 treatment
#   "blood_pressure_medication_2",
#
#   # Outcome
#   "mortality"
# )]
#
# # Correct order?
# Lnodes <- c(
#   # Baseline covariates
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
```

```
#   "bmi", "blood_pressure", "chol",
#
#   # Time 2 measurements (these come after time 1 treatment)
#   "bmi_2", "blood_pressure_2", "chol_2"
# )
#
# Anodes <- c("blood_pressure_medication", "blood_pressure_medication_2")
# Ynodes <- "mortality"
#
# # Run full LTMLE with proper ordering
# full_ltmle_result <- ltmle(
#   data = data_ordered, # Use the ordered dataset
#   Anodes = Anodes,
#   Lnodes = Lnodes,
#   Ynodes = Ynodes,
#   abar = list(c(1,1), c(0,0)),
#   SL.library = SL.library
# )
#
# summary(full_ltmle_result)
```

Ok I think R was crashing because of the amount of data, not the library, so I'm going to run the code with a subset of 1000 and the bigger library.

update – 500 seems to be about what it can handle in a normal amount of time. commenting other attempts out

```
#21
## Subset data -- larger subset
#
#
#
# # Define SuperLearner library (same as)
# SL.library <- c(
#   'SL.mean',    # Baseline
#   'SL.glm',     # Logistic regression
#   'SL.glmnet',  # Lasso and ridge
#   'SL.ranger',  # Random forest
#   'SL.xgboost'  # Gradient boosting
# )
#
#
# # Ltmle nodes order:
# # 1. Baseline covariates (Lnodes)
# # 2. Time 1 treatment (Anodes)
# # 3. Time 2 covariates (Lnodes)
# # 4. Time 2 treatment (Anodes)
# # 5. Outcome (Ynodes)
#
# set.seed(123)
#
```

```
# # Take a random sample of 1000 observations
# data_subset <- heart_disease[sample(nrow(heart_disease), 1000), ]
#
#
# # order
# data_ordered <- data_subset[, c(
#   # Baseline covariates and baseline measurements
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
#   "bmi", "blood_pressure", "chol",
#
#   # Time 1 treatment
#   "blood_pressure_medication",
#
#   # Time 2 measurements
#   "bmi_2", "blood_pressure_2", "chol_2",
#
#   # Time 2 treatment
#   "blood_pressure_medication_2",
#
#   # Outcome
#   "mortality"
# )]
#
# # Correct order?
# Lnodes <- c(
#   # Baseline covariates
#   "age", "sex_at_birth", "simplified_race", "college_educ", "income_thousands",
#   "bmi", "blood_pressure", "chol",
#
#   # Time 2 measurements (these come after time 1 treatment)
#   "bmi_2", "blood_pressure_2", "chol_2"
# )
#
# Anodes <- c("blood_pressure_medication", "blood_pressure_medication_2")
# Ynodes <- "mortality"
#
# # Run full LTMLE with proper ordering
# full_ltmle_result <- ltmle(
#   data = data_ordered, # Use the ordered dataset
#   Anodes = Anodes,
#   Lnodes = Lnodes,
#   Ynodes = Ynodes,
#   abar = list(c(1,1), c(0,0)),
#   SL.library = SL.library
# )
#
# summary(full_ltmle_result)
```

results from naive estimate

ltmle(data = naive_data, Anodes = “A”, Lnodes = NULL, Ynodes = “Y”, abar = list(1, 0), SL.library = SL.library)

Treatment Estimate: Parameter Estimate: 0.52163 Estimated Std Err: 0.013638 p-value: <2e-16 95% Conf Interval: (0.4949, 0.54837)

Control Estimate: Parameter Estimate: 0.51625 Estimated Std Err: 0.0053737 p-value: <2e-16 95% Conf Interval: (0.50571, 0.52678)

Additive Treatment Effect: Parameter Estimate: 0.0053896 Estimated Std Err: 0.014659 p-value: 0.71312 95% Conf Interval: (-0.023341, 0.03412)

Relative Risk: Parameter Estimate: 1.0104 Est Std Err log(RR): 0.028141 p-value: 0.71208 95% Conf Interval: (0.95622, 1.0677)

Odds Ratio: Parameter Estimate: 1.0218 Est Std Err log(OR): 0.058739 p-value: 0.71321 95% Conf Interval: (0.91071, 1.1465)

Naive: Outcomes look worse in treatment than control (52.16% vs 51.63%), but difference is not statistically significant, which is good because we don't want to think that the medication is actually harmful!

Results with 500 and full library

Call: ltmle(data = data_ordered, Anodes = Anodes, Lnodes = Lnodes, Ynodes = Ynodes, abar = list(c(1, 1), c(0, 0)), SL.library = SL.library)

Treatment Estimate: Parameter Estimate: 0.21251 Estimated Std Err: 0.14686 p-value: 0.14789 95% Conf Interval: (0, 0.50034)

Control Estimate: Parameter Estimate: 0.59643 Estimated Std Err: 0.026403 p-value: <2e-16 95% Conf Interval: (0.54468, 0.64818)

Additive Treatment Effect: Parameter Estimate: -0.38393 Estimated Std Err: 0.14921 p-value: 0.010081 95% Conf Interval: (-0.67638, -0.091478)

Relative Risk: Parameter Estimate: 0.3563 Est Std Err log(RR): 0.69249 p-value: 0.13615 95% Conf Interval: (0.091699, 1.3844)

Odds Ratio: Parameter Estimate: 0.18259 Est Std Err log(OR): 0.88438 p-value: 0.054503 95% Conf Interval: (0.032262, 1.0334)

Results with 1000 observations and smaller library

Treatment Estimate: Parameter Estimate: 0.28869 Estimated Std Err: 0.11583 p-value: 0.012689 95% Conf Interval: (0.061668, 0.5157)

Control Estimate: Parameter Estimate: 0.58878 Estimated Std Err: 0.018992 p-value: <2e-16 95% Conf Interval: (0.55156, 0.626)

Additive Treatment Effect: Parameter Estimate: -0.3001 Estimated Std Err: 0.11737 p-value: 0.010565 95% Conf Interval: (-0.53014, -0.070048)

Relative Risk: Parameter Estimate: 0.49031 Est Std Err log(RR): 0.40252 p-value: 0.076619 95% Conf Interval: (0.22277, 1.0792)

Odds Ratio: Parameter Estimate: 0.28345 Est Std Err log(OR): 0.56949 p-value: 0.026845 95% Conf Interval: (0.09284, 0.86542)

Results with 1000 observations and full library (took 5 hours, commenting out)

Call: ltmle(data = data_ordered, Anodes = Anodes, Lnodes = Lnodes, Ynodes = Ynodes, abar = list(c(1, 1), c(0, 0)), SL.library = SL.library)

Treatment Estimate: Parameter Estimate: 0.2741 Estimated Std Err: 0.09844 p-value: 0.0053622 95% Conf Interval: (0.08116, 0.46704)

Control Estimate: Parameter Estimate: 0.5896 Estimated Std Err: 0.018418 p-value: <2e-16 95% Conf Interval: (0.5535, 0.62569)

Additive Treatment Effect: Parameter Estimate: -0.3155 Estimated Std Err: 0.10015 p-value: 0.0016309 95% Conf Interval: (-0.51178, -0.11921)

Relative Risk: Parameter Estimate: 0.46489 Est Std Err log(RR): 0.3605 p-value: 0.03361 95% Conf Interval: (0.22935, 0.94234)

Odds Ratio: Parameter Estimate: 0.26284 Est Std Err log(OR): 0.50057 p-value: 0.0075988 95% Conf Interval: (0.098537, 0.70109)

The results of the LTLMLE with 1,000 observations and the full SL library and two time-points shows a positive impact of treatment (31.55 percentage points) that's statistically significant. P values are nicely small, confidence intervals don't include zero, risk and odds and additive effect all point towards a real treatment effect.

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 is worrisome because a treatment can change a variable (e.g. blood pressure), and then that variable can in turn affects future treatments (and outcomes). So some variables are both influenced by earlier treatments and also impact future ones. This means that we have to be careful about adjusting for them, trying not to totally erase their influence but also not allowing them to confound. Unlike things like blood pressure that change quickly and in unexpected ways, age isn't a confounder because it changes steadily and isn't affected by treatment.