Project 8 Template

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

#heart_disease <- read_csv('/Users/alexadia/Documents/GitHub/Computational-Social-Science-Projects/Proj
heart_disease <- read_csv('D:/GitHub/Computational-Social-Science-Projects/Project 8/heart_disease_tmle
```

```
## Rows: 10000 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): age, sex_at_birth, simplified_race, college_educ, income_thousands...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Introduction

Heart disease is the leading cause of death in the United States, and treating it properly is an important public health goal. However, it is a complex disease with several different risk factors and potential treatments. Physicians typically recommend changes in diet, increased exercise, and/or medication to treat symptoms, but it is difficult to determine how effective any one of these factors is in treating the disease. In this project, you will explore SuperLearner, Targeted Maximum Likelihood Estimation (TMLE), and Longitudinal Targeted Maximum Likelihood Estimation (LTMLE). Using a simulated dataset, you will explore whether taking blood pressure medication reduces mortality risk.

Data

This dataset was simulated using R (so it does not come from a previous study or other data source). It contains several variables:

- **blood_pressure_medication**: Treatment indicator for whether the individual took blood pressure medication (0 for control, 1 for treatment)

- **mortality**: Outcome indicator for whether the individual passed away from complications of heart disease (0 for no, 1 for yes)
- **age**: Age at time 1
- **sex_at_birth**: Sex assigned at birth (0 female, 1 male)
- **simplified_race**: Simplified racial category. (1: White/Caucasian, 2: Black/African American, 3: Latinx, 4: Asian American,
  5: Mixed Race/Other)
- **income_thousands**: Household income in thousands of dollars
- **college_educ**: Indicator for college education (0 for no, 1 for yes)
- **bmi**: Body mass index (BMI)
- **chol**: Cholesterol level
- **blood_pressure**: Systolic blood pressure
- **bmi_2**: BMI measured at time 2
- **chol_2**: Cholesterol measured at time 2
- **blood_pressure_2**: BP measured at time 2
- **blood_pressure_medication_2**: Whether the person took treatment at time period 2

For the "SuperLearner" and "TMLE" portions, you can ignore any variable that ends in "_2", we will reintroduce these for LTMLE.

SuperLearner

Modeling

Fit a SuperLearner model to estimate the probability of someone dying from complications of heart disease, conditional on treatment and the relevant covariates. Do the following:

1. Choose a library of at least 5 machine learning algorithms to evaluate. **Note**: We did not cover how to hyperparameter tune constituent algorithms within SuperLearner in lab, but you are free to do so if you like (though not required to for this exercise).
2. Split your data into train and test sets.
3. Train SuperLearner
4. Report the risk and coefficient associated with each model, and the performance of the discrete winner and SuperLearner ensemble
5. Create a confusion matrix and report your overall accuracy, recall, and precision

```
# Fit SuperLearner Model

## sl lib
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"
```

```
library<-c('SL.mean', 'SL.glmnet', 'SL.earth', "SL.lm", "SL.nnet") #picked these arbitrarily

## Train/Test split
# need to kick the variables that start with 2 since we ignore for this section
heart_disease_no2<-heart_disease%>%select(-bmi_2, -chol_2, -blood_pressure_2, -blood_pressure_medicatio
#now get split
set.seed(24)
heart_disease_split<-initial_split(heart_disease_no2)
# Declare the training set with rsample::training()
train <- training(heart_disease_split)

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

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

# Do the same procedure with the test set
test <- testing(heart_disease_split)

y_test <- test %>%
  pull(mortality)

x_test <- test %>%
  select(-mortality)

## Train SuperLearner
```

```r
sl_q1<- SuperLearner(Y = y_train,
                         X = x_train,
                         family = binomial(),
                         SL.library = library)
```

```
## Loading required namespace: earth
```

```r
## Risk and Coefficient of each model
sl_q1
```

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = library)
##
##
##
##                     Risk      Coef
## SL.mean_All    0.2498864 0.0000000
## SL.glmnet_All 0.2364546 0.0000000
## SL.earth_All   0.2300872 0.8484226
## SL.lm_All      0.2361808 0.1515774
## SL.nnet_All    0.2504166 0.0000000
```

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

## Confusion Matrix

preds <- predict(sl_q1,
                 x_test,
                 onlySL = TRUE)

# start with y_test
validation <- y_test %>%
  # add our predictions
  bind_cols(preds$pred[,1]) %>%
  # rename columns
  rename(obs = `...1`,
         pred = `...2`) %>%
  mutate(pred = ifelse(pred >= .5,
                           1,
                           0))
```

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

```r
head(validation)
```

```
## # A tibble: 6 x 2
##     obs  pred
##   <dbl> <dbl>
```

```
## 1     1     1
## 2     0     1
## 3     0     1
## 4     1     1
## 5     0     1
## 6     0     1
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  257   66
##          1  920 1257
##
##                Accuracy : 0.6056
##                  95% CI : (0.5861, 0.6248)
##     No Information Rate : 0.5292
##     P-Value [Acc > NIR] : 8.566e-15
##
##                   Kappa : 0.1755
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.2184
##             Specificity : 0.9501
##          Pos Pred Value : 0.7957
##          Neg Pred Value : 0.5774
##              Prevalence : 0.4708
##          Detection Rate : 0.1028
##    Detection Prevalence : 0.1292
##       Balanced Accuracy : 0.5842
##
##        'Positive' Class : 0
##
```

Discussion Questions

1. Why should we, in general, prefer the SuperLearner ensemble to the discrete winner in cross-validation? Or in other words, what is the advantage of "blending" algorithms together and giving them each weights, rather than just using the single best algorithm (with best being defined as minimizing risk)?

2. Answer: SuperLearner allows us to avoid the challenge of having to pick a single "best" algorithm by using a suite of them to get to the best estimate - even if there is a clear, dominant and best algorithm, if it is included in the SuperLearner, then that algorithm dominates and you get the same results as if you had used that "best" algorithm alone (assuming a sufficiently large sample).

Targeted Maximum Likelihood Estimation

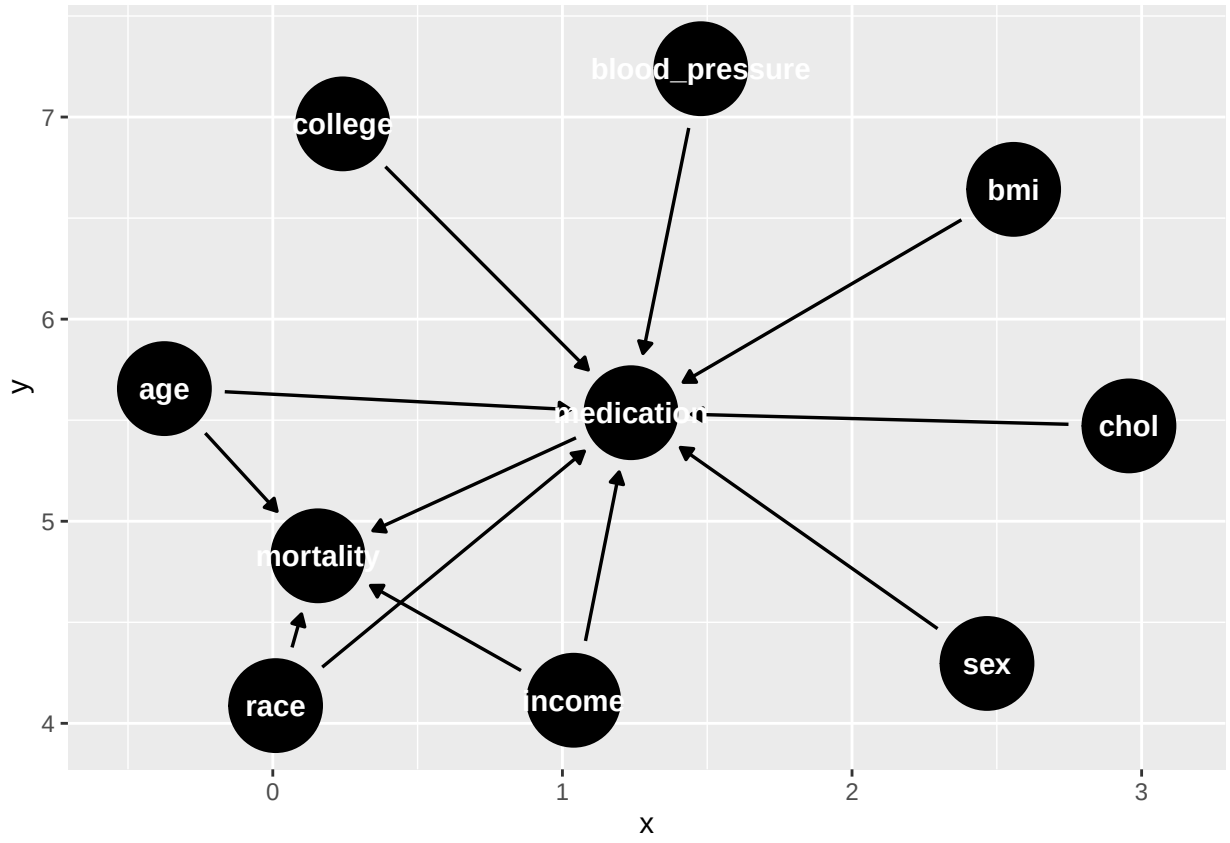
Causal Diagram

TMLE requires estimating two models:

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

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

```
# DAG for TMLE
tmle_dag<-dagify(medication ~ age + race + college + chol + bmi+ income + sex + blood_pressure,
                 mortality ~ medication + age + income + race,
                 exposure = "medication",
                 outcome="mortality"
)
ggdag(tmle_dag)
```



```
#pretty_dag(tmle_dag) - didnt work but maybe could look back
```

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

```
#define models
Y<-heart_disease_no2$mortality
A<-heart_disease_no2$blood_pressure_medication
W<-heart_disease_no2%>%select(age, income_thousands, simplified_race)

tmle_result <- tmle(Y=Y,
                    A=A,
                    W=W,
                    family="binomial",
                    Q.SL.library = library,
                    g.SL.library = library)

summary(tmle_result)
```

```
##  Initial estimation of Q
##   Procedure: cv-SuperLearner, ensemble
##   Model:
##       Y ~  SL.mean_All + SL.glmnet_All + SL.earth_All + SL.lm_All + SL.nnet_All
##
##   Coefficients:
##       SL.mean_All    0
##     SL.glmnet_All    0.5335044
##      SL.earth_All    0
##         SL.lm_All    0.4664956
##       SL.nnet_All    0
##
##   Cross-validated pseudo R squared :  0.0383
##
##  Estimation of g (treatment mechanism)
##   Procedure: SuperLearner, ensemble
##   Model:
##       A ~  SL.mean_All + SL.glmnet_All + SL.earth_All + SL.lm_All + SL.nnet_All
##
##   Coefficients:
##       SL.mean_All    0.02720436
##     SL.glmnet_All    0
##      SL.earth_All    0.9727956
```

```
##         SL.lm_All    0
##       SL.nnet_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:  0.24626
##    Estimated Variance:  0.00012388
##                p-value:  <2e-16
##     95% Conf Interval:  (0.22444, 0.26807)
##
##  Marginal Mean under Comparator (EY0)
##    Parameter Estimate:  0.56588
##    Estimated Variance:  2.912e-05
##                p-value:  <2e-16
##     95% Conf Interval:  (0.55531, 0.57646)
##
##  Additive Effect
##    Parameter Estimate:  -0.31963
##    Estimated Variance:  0.00015294
##                p-value:  <2e-16
##     95% Conf Interval:  (-0.34387, -0.29539)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.31653
##    Estimated Variance:  0.00015196
##                p-value:  <2e-16
##     95% Conf Interval:  (-0.34069, -0.29237)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.32022
##    Estimated Variance:  0.0001549
##                p-value:  <2e-16
##     95% Conf Interval:  (-0.34462, -0.29583)
##
##  Relative Risk
##    Parameter  Estimate:  0.43517
##    Variance(log scale):  0.0021332
##                 p-value:  <2e-16
##      95% Conf Interval:  (0.39751, 0.4764)
##
##  Odds Ratio
##     Parameter  Estimate:  0.25064
##     Variance(log scale):  0.0040768
##                  p-value:  <2e-16
##       95% Conf Interval:  (0.22115, 0.28405)
```

```
tmle_result$estimates$ATE
```

```
## $psi
## [1] -0.3196268
##
## $var.psi
## [1] 0.0001529355
##
## $CI
## [1] -0.3438651 -0.2953885
##
## $pvalue
## [1] 2.716316e-147
##
## $bs.var
## [1] NA
##
## $bs.CI.twosided
## [1] NA NA
##
## $bs.CI.onesided.lower
## [1] -Inf   NA
##
## $bs.CI.onesided.upper
## [1]  NA Inf
```

Discussion Questions

1. What is a "double robust" estimator? Why does it provide a guarantee of consistency if either the outcome model or propensity score model is correctly specified? Or in other words, why does mispecifying one of the models not break the analysis? **Hint**: When answering this question, think about how your introductory statistics courses emphasized using theory to determine the correct outcome model, and in this course how we explored the benefits of matching.

2. Answer: A doubly robust estimator combines two approaches to estimating the causal model, which is helpful in avoiding the consequences of model misspecification using a single approach. With the doubly robust estimator, even if one approach is misspecified, we can get a consistent estimator of the effect as the estimator will upweight the outputs of the correctly specified approach. If there is no misspecification, our estimator is as efficient as just using the single (correct approach.

LTMLE Estimation

Now imagine that everything you measured up until now was in “time period 1”. Some people either choose not to or otherwise lack access to medication in that time period, but do start taking the medication in time period 2. Imagine we measure covariates like BMI, blood pressure, and cholesterol at that time for everyone in the study (indicated by a “_2” after the covariate name).

Causal Diagram

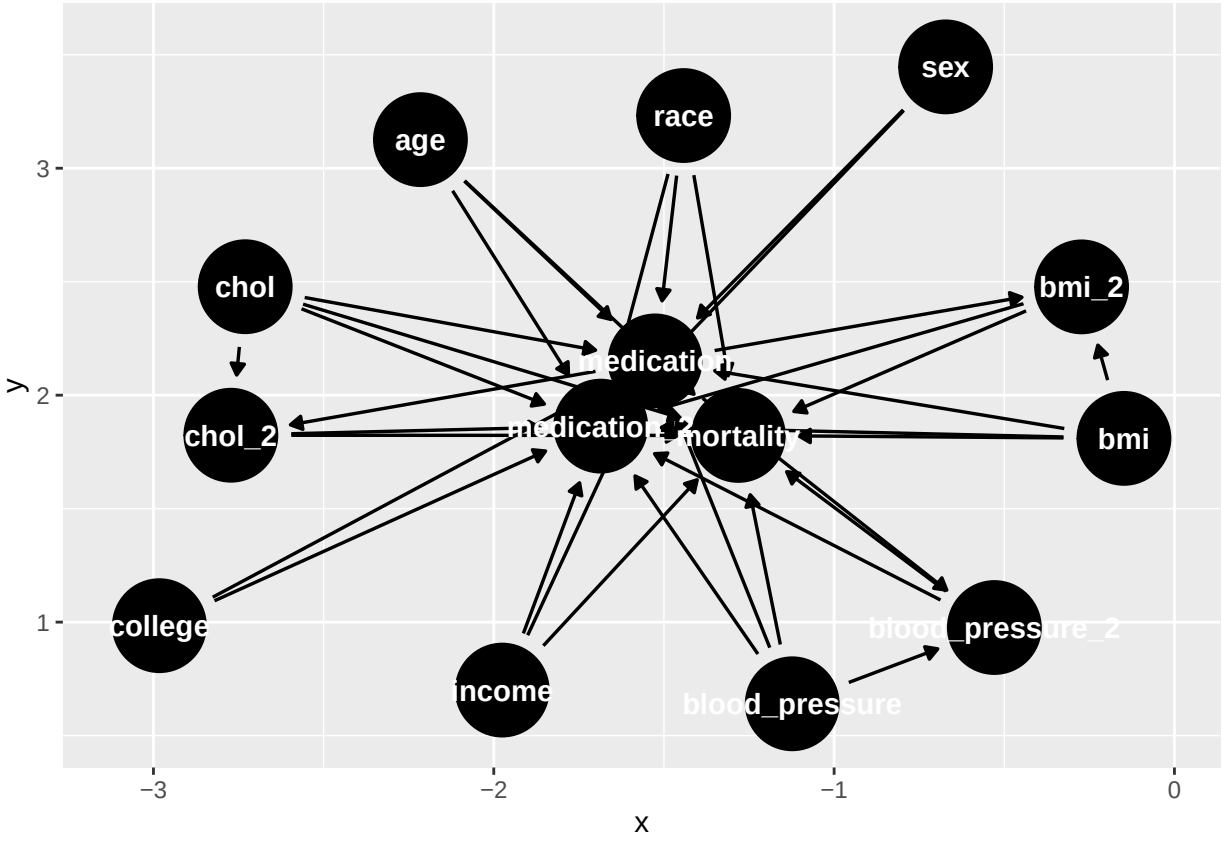
Update your causal diagram to incorporate this new information. **Note**: If your groups divides up sections and someone is working on LTMLE separately from TMLE then just draw a causal diagram even if it does

not match the one you specified above.

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

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

```
# DAG for TMLE
ltmle_dag<-dagify(medication ~ age + race + college + chol + bmi+ income + sex + blood_pressure,
                  mortality ~ medication + age + income + race+chol+bmi+blood_pressure+blood_pressure_2+
                  blood_pressure_2~blood_pressure+medication,
                  chol_2~chol+medication,
                  bmi_2~bmi+medication,
                  medication_2~ age + race + college + chol + bmi+ income + sex + blood_pressure+blood_p
                  exposure = c("medication", "medication_2"),
                  outcome="mortality"
)
ggdag(ltmle_dag) #sorry Kasey this is a chaotic mess
```



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?

```r
#LTMLE setup
#reorder dataset in order of W, A1, L, A2, Y
ltmle_data<-heart_disease%>%select(age, simplified_race, income_thousands, chol, bmi, blood_pressure, bl
#define nodes - don't think you need to define W because it just assumes whatever is to the left of A1
Anodes<-c("blood_pressure_medication", "blood_pressure_medication_2")
Ynodes<-"mortality"
Lnodes<-c("bmi_2", "chol_2", "blood_pressure_2")

## Naive Model (no time-dependent confounding) estimate
ltmle_naive_data<-ltmle_data%>%select(-bmi_2, -chol_2, -blood_pressure_2)
ltmle_naive<-ltmle(ltmle_naive_data, Anodes=Anodes, Ynodes=Ynodes, abar=c(1, 1), SL.library=library)
```

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

## formula for mortality:

## Q.kplus1 ~ age + simplified_race + income_thousands + chol +     bmi + blood_pressure + blood_pressu

##

## gform not specified, using defaults:

## formula for blood_pressure_medication:

## blood_pressure_medication ~ age + simplified_race + income_thousands +     chol + bmi + blood_pressu

## formula for blood_pressure_medication_2:

## blood_pressure_medication_2 ~ age + simplified_race + income_thousands +     chol + bmi + blood_press

##

## 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

## Estimate of time to completion: 3 to 7 minutes
```

```r
summary(ltmle_naive)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = ltmle_naive_data, Anodes = Anodes, Ynodes = Ynodes,
##     abar = c(1, 1), SL.library = library)
##
##    Parameter Estimate:  0.19679
##     Estimated Std Err:  0.029036
##               p-value:  1.2235e-11
##     95% Conf Interval: (0.13988, 0.2537)
```

```
## LTMLE estimate
ltmle<-ltmle(ltmle_data, Anodes=Anodes, Lnodes=Lnodes, Ynodes=Ynodes, abar=c(1, 1), SL.library = library
```

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

## formula for bmi_2:

## Q.kplus1 ~ age + simplified_race + income_thousands + chol +     bmi + blood_pressure + blood_pressu

## formula for mortality:

## Q.kplus1 ~ age + simplified_race + income_thousands + chol +     bmi + blood_pressure + blood_pressu

##

## gform not specified, using defaults:

## formula for blood_pressure_medication:

## blood_pressure_medication ~ age + simplified_race + income_thousands +     chol + bmi + blood_pressu

## formula for blood_pressure_medication_2:

## blood_pressure_medication_2 ~ age + simplified_race + income_thousands +     chol + bmi + blood_pres

##

## Estimate of time to completion: 3 to 9 minutes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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
```

```
## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## 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

## Warning in FUN(X[[i]], ...): Error in algorithm SL.glmnet  on full data
##   The Algorithm will be removed from the Super Learner (i.e. given weight 0)

## Warning in SuperLearner::SuperLearner(Y = Y.subset, X = X.subset, SL.library =
## SL.library, : Coefficients already 0 for all failed algorithm(s)
```

```
summary(ltmle)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = ltmle_data, Anodes = Anodes, Lnodes = Lnodes, Ynodes = Ynodes,
##     abar = c(1, 1), SL.library = library)
##
##    Parameter Estimate:  0.20488
##     Estimated Std Err:  0.029844
##               p-value:  6.6486e-12
##     95% Conf Interval: (0.14639, 0.26337)
```

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?

2. Answer: We should be most worried about confounders who can experience rapid changes over the given time period of measurement in a way that may impact the exposure-outcome relationship. In the example given, becoming incremently older is unlikely to be very impactful unless a patient is at the Medicare cutoff (i.e., is about to become 65); however, blood pressure can change rapidly over time if, for example, heart failure or a myocardial infarction is approaching, making it more important to address time-dependent confounding. Ultimately, drawing the DAG and making your assumptions about the relationship clear is most important.