Project 8 Template

```r
# Add to this package list for additional SL algorithms
pacman::p_load(
  tidyverse,
  ggthemes,
  ltmle,
  tmle,
  SuperLearner,
  tidymodels,
  caret,
  dagitty,
  ggdag,
  here,
  glmnet,
  randomForest,
  xgboost,
  ranger)
setwd("~/git/Computational-Social-Science-Projects/Project 8")
heart_disease <- read_csv('heart_disease_tmle.csv')
```

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

Introduction

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

Data

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

- **blood_pressure_medication**: Treatment indicator for whether the individual took blood pressure medication (0 for control, 1 for treatment)
- **mortality**: Outcome indicator for whether the individual passed away from complications of heart disease (0 for no, 1 for yes)

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

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

SuperLearner

Modeling

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

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

```r
# Fit SuperLearner Model

## sl lib
sl_lib <- c("SL.glmnet", "SL.ranger", "SL.xgboost", "SL.gam", "SL.randomForest")
## Train/Test split
set.seed(123)
heart_disease_split <- initial_split(heart_disease, prop = 0.8)
heart_disease_train <- training(heart_disease_split)
heart_disease_test <- testing(heart_disease_split)

y_train <- heart_disease_train$mortality
x_train <- heart_disease_train %>% select(-mortality, -matches("_2$"))

y_test <- heart_disease_test$mortality
```

```r
x_test <- heart_disease_test %>% select(-mortality, -matches("_2$"))


## Create SuperLearner object
## Train SuperLearner
sl <- SuperLearner(Y = y_train,
                    X = x_train,
                    SL.library = sl_lib,
                    family = binomial(),
                    verbose = TRUE,
                    method = "method.NNLS",
                    cvControl = list(V = 5))
```

```
## Number of covariates in All is: 9

## CV SL.glmnet_All

## CV SL.ranger_All

## CV SL.xgboost_All

## CV SL.gam_All

## CV SL.randomForest_All

## Number of covariates in All is: 9

## CV SL.glmnet_All

## CV SL.ranger_All

## CV SL.xgboost_All

## CV SL.gam_All

## CV SL.randomForest_All

## Number of covariates in All is: 9

## CV SL.glmnet_All

## CV SL.ranger_All

## CV SL.xgboost_All

## CV SL.gam_All

## CV SL.randomForest_All

## Number of covariates in All is: 9

## CV SL.glmnet_All

## CV SL.ranger_All

## CV SL.xgboost_All

## CV SL.gam_All

## CV SL.randomForest_All

## Number of covariates in All is: 9

## CV SL.glmnet_All

## CV SL.ranger_All
## CV SL.xgboost_All

## CV SL.gam_All

## CV SL.randomForest_All

## Non-Negative least squares convergence: TRUE

## full SL.glmnet_All

## full SL.ranger_All

## full SL.xgboost_All

## full SL.gam_All

## full SL.randomForest_All
```

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

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = sl_lib,
##     method = "method.NNLS", verbose = TRUE, cvControl = list(V = 5))
##
##
##                          Risk       Coef
## SL.glmnet_All       0.2352450 0.38788323
## SL.ranger_All       0.2321326 0.03971521
## SL.xgboost_All      0.2505555 0.00000000
## SL.gam_All          0.2355384 0.00000000
## SL.randomForest_All 0.2317225 0.57240157
```

```
## Discrete winner and superlearner ensemble performance
discrete_winner <- sl$libraryNames[which.min(sl$cvRisk)]
cat("Discrete Super Learner (best algorithm):", discrete_winner, "\n")
```

```
## Discrete Super Learner (best algorithm): SL.randomForest_All
```

```
## Confusion Matrix
preds <- predict(sl, x_test, onlySL = TRUE)


validation <- heart_disease_test %>%
  mutate(pred = preds$pred[,1]) %>%
  mutate(pred = ifelse(pred > 0.5, 1, 0)) %>%
  select(mortality, pred)

confusion <- confusionMatrix(as.factor(validation$pred), as.factor(validation$mortality))

confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 307 150
##          1 713 830
##
##                Accuracy : 0.5685
##                   95% CI : (0.5465, 0.5903)
##      No Information Rate : 0.51
##      P-Value [Acc > NIR] : 8.911e-08
##
##                    Kappa : 0.1463
##
##   Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.3010
##              Specificity : 0.8469
##           Pos Pred Value : 0.6718
##           Neg Pred Value : 0.5379
##               Prevalence : 0.5100
##           Detection Rate : 0.1535
##     Detection Prevalence : 0.2285
##        Balanced Accuracy : 0.5740
##
##         'Positive' Class : 0
##
```

```r
accuracy <- confusion$overall["Accuracy"]
recall <- confusion$byClass["Sensitivity"]
precision <- confusion$byClass["Pos Pred Value"]

cat("Accuracy :", round(accuracy, 4), "\n")
```

```
## Accuracy : 0.5685
```

```r
cat("Recall   :", round(recall, 4), "\n")
```

```
## Recall   : 0.301
```

```r
cat("Precision:", round(precision, 4), "\n")
```

```
## Precision: 0.6718
```

Discussion Questions

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

The discrete winner in cross-validation performed best on average in the training sample, but could still be prone to overfitting in small samples or noisy data. The SuperLearner ensemble, on the other hand, combines the strengths of multiple algorithms and can provide a more robust estimate of the outcome. By blending algorithms together, we can reduce the risk of overfitting and improve generalization to new data as the blended algorithms will better capture different patterns in the data that can be traded off with individual algorithms. So, SuperLearner performs at least as well as the best weighted average of candidate algorithms and better than any single algorithm.

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, $P(Y|(A, W)$.

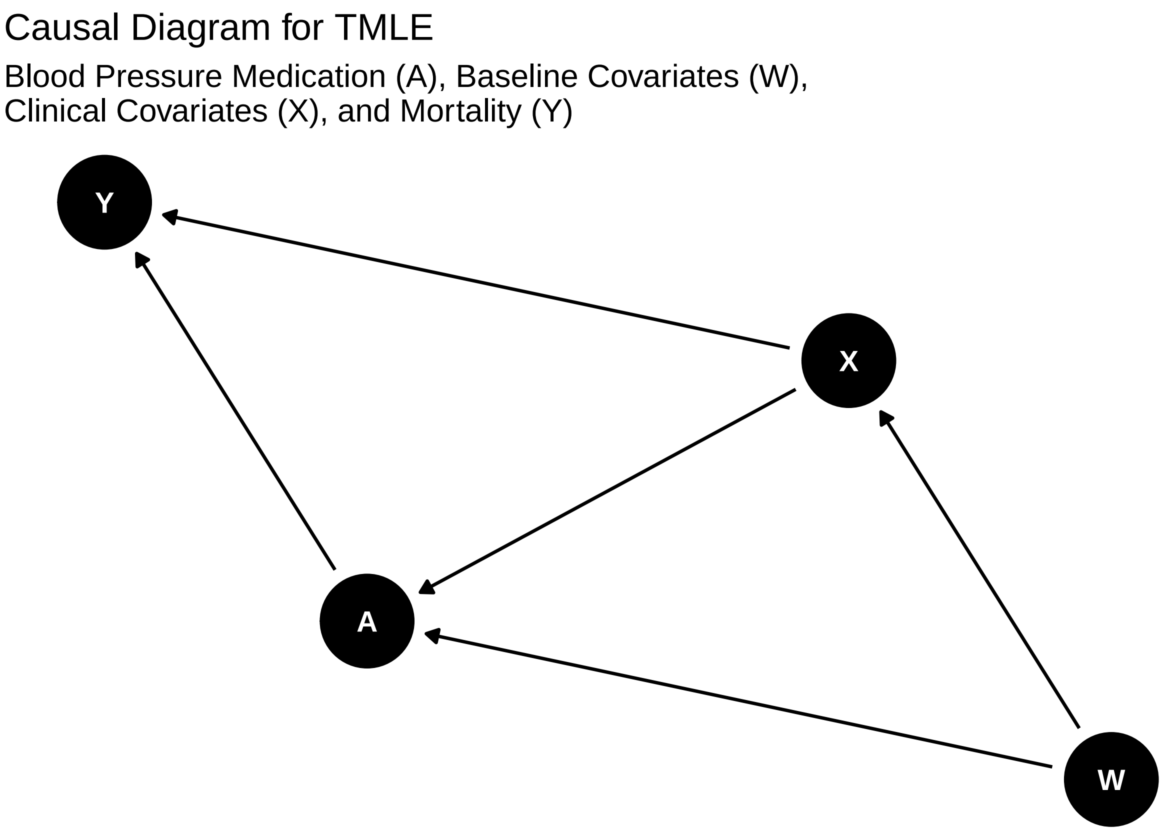
2. The propensity score model, or the relationship between assignment to treatment and predictors $P(A|W)$

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.

```r
# DAG for TMLE


dag <- dagitty("dag {
  A [exposure, label=\"Treatment\"]
  Y [outcome, label=\"Outcome\"]
  W [label=\"Baseline Covariates\"]
  X [label=\"Clinical Covariates\"]
  W -> A
  W -> X
  X -> A
  A -> Y
  X -> Y
}")

ggdag(dag) +
  theme_dag() +
  theme(legend.position = "none") +
  labs(title = "Causal Diagram for TMLE",
       subtitle = "Blood Pressure Medication (A), Baseline Covariates (W), \nClinical Covariates (X), a
```


Causal Diagram for TMLE
Blood Pressure Medication (A), Baseline Covariates (W),
Clinical Covariates (X), and Mortality (Y)
Y
X
A
W

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

```
# TMLE Estimation

covariates <- heart_disease %>%
  select(-matches("_2$")) %>%
  select(-mortality, -blood_pressure_medication)

library(doParallel)
```

```
## Loading required package: iterators

## Loading required package: parallel
```

```
n_cores <- parallel::detectCores() - 1
cl      <- makeCluster(n_cores)
registerDoParallel(cl)



tmle_fit <- tmle(Y = heart_disease$mortality,
                 A = heart_disease$blood_pressure_medication,
                 W = covariates,
                 family = "binomial",
                 Q.SL.library = sl_lib,
                 g.SL.library = sl_lib,
                 verbose = TRUE)
```

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

```
stopCluster(cl)
registerDoSEQ()

#Reporting
tmle_summary <- summary(tmle_fit)

# Average treatment effect (ATE)
ate <- tmle_summary$estimates$ATE$psi

# Standard error
se <- tmle_summary$estimates$ATE$std.err

# 95% confidence interval
ci_lower <- tmle_summary$estimates$ATE$CI[1]
```

```r
ci_upper <- tmle_summary$estimates$ATE$CI[2]

# p-value
p_val <- tmle_summary$estimates$ATE$pvalue

# Report results
cat(sprintf("Average Treatment Effect (ATE): %.4f\n", ate))
```

```
## Average Treatment Effect (ATE): -0.3550
```

```r
cat(sprintf("95%% Confidence Interval: [%.4f, %.4f]\n", ci_lower, ci_upper))
```

```
## 95% Confidence Interval: [-0.3717, -0.3384]
```

```r
cat(sprintf("Standard Error: %.4f\n", se))
cat(sprintf("P-value: %.4f\n", p_val))
```

```
## P-value: 0.0000
```

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.

A double robust estimator is a type of estimator that remains consistent if either the outcome model or the propensity score model is correctly specified, but not necessarily both. This means that if one of the models is correctly specified, the estimator will still provide valid estimates of the treatment effect. This is the case because it combines the strengths of matching and regression adjustment. The estimation procedure combines information from both models: If the outcome model is correctly specified, the treatment effect estimator will be consistent, and the propensity score model provides additional efficiency gains by better leveraging variation in the data. If the propensity score model is correctly specified, it will adjust for confounding such that the outcome model doesn't dominate the effects. This property allows for greater flexibility in model specification and reduces the risk of bias due to model misspecification.

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.

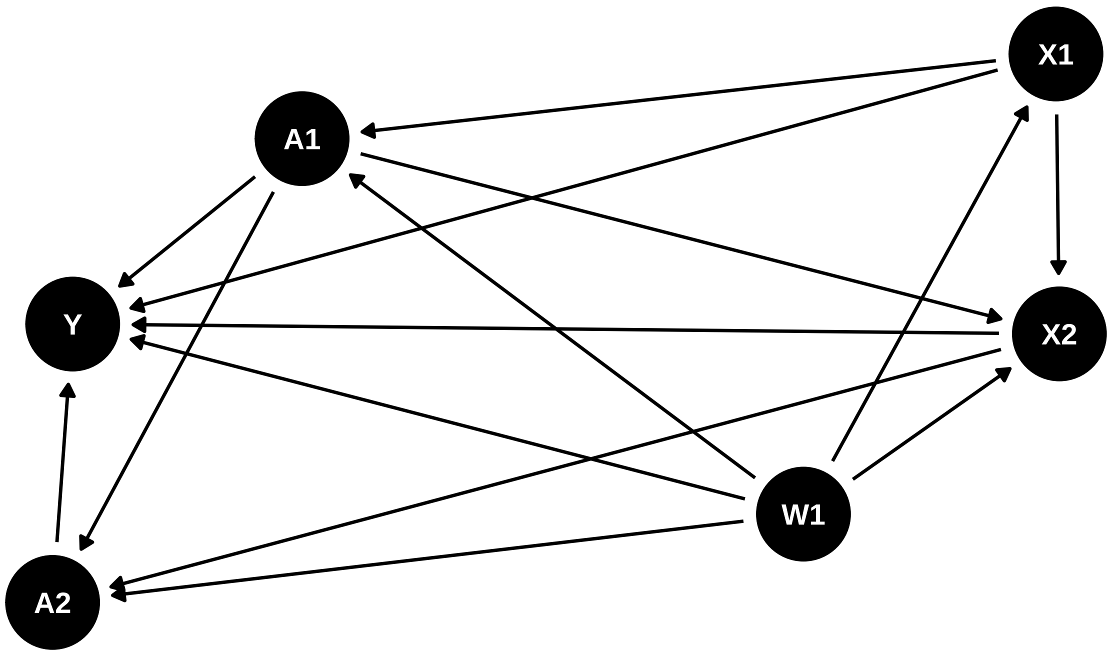
```
# DAG for LTMLE

dag2 <- dagitty("dag {
  A1 [exposure, label=\"Treatment 1\"]
  A2 [exposure, label=\"Treatment 2\"]
  Y [outcome, label=\"Outcome\"]
  W1 [label=\"Baseline Covariates 1\"]
  X1 [label=\"Clinical Covariates 1\"]
  X2 [label=\"Clinical Covariates 2\"]
  W1 -> X1
  W1 -> X2
  W1 -> A1
  W1 -> A2
  W1 -> Y
  X1 -> X2
  X1 -> Y
  X2 -> Y
  X1 -> A1
  X2 -> A2
  A1 -> A2
  A1 -> Y
  A2 -> Y
  A1 -> X2
}")

ggdag(dag2) +
  theme_dag() +
  theme(legend.position = "none") +
  labs(title = "Causal Diagram for LTMLE",
       subtitle = "Blood Pressure Medication (A), Baseline Covariates (W), \nClinical Covariates (X), a
```

Causal Diagram for LTMLE

Blood Pressure Medication (A), Baseline Covariates (W),
Clinical Covariates (X), and Mortality (Y),
1 corresponds to time period 1, 2 corresponds to time period 2



LTMLE Estimation

Use the `ltmle` package for this section. First fit a “naive model” that **does not** control for the time-dependent confounding. Then run a LTMLE model that does control for any time dependent confounding. Follow the same steps as in the TMLE section. Do you see a difference between the two estimates?

```
## Naive Model (no time-dependent confounding) estimate
# This model ignores all possible confounding

library(future)
```

```
##
## Attaching package: 'future'

## The following object is masked from 'package:caret':
##
##     cluster
```

```
library(future.apply)

plan(multisession, workers = parallel::detectCores() - 1)  # leave 1 core free
plan(multicore, workers = parallel::detectCores() - 1)
```

```
## Warning in supportsMulticoreAndRStudio(...): [ONE-TIME WARNING] Forked
## processing ('multicore') is not supported when running R from RStudio because
## it is considered unstable. For more details, how to control forked processing
## or not, and how to silence this warning in future R sessions, see
## ?parallelly::supportsMulticore
```

```r
data_ltmle_naive <- heart_disease %>%
  rename(A1 = blood_pressure_medication,
         A2 = blood_pressure_medication_2,
         Y = mortality) %>%
  select(A1, A2, Y)

ltmle_naive <- ltmle(
  data = data_ltmle_naive,
  Anodes = c("A1", "A2"),
  Ynodes = "Y",
  survivalOutcome = FALSE,
  abar = c(1, 1),
  SL.library = sl_lib
)
```

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

## formula for Y:

## Q.kplus1 ~ A1 + A2

##

## gform not specified, using defaults:

## formula for A1:

## A1 ~ 1

## formula for A2:

## A2 ~ A1

##

## 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
# Reporting
summary(ltmle_naive)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = data_ltmle_naive, Anodes = c("A1", "A2"), Ynodes = "Y",
##     survivalOutcome = FALSE, abar = c(1, 1), SL.library = sl_lib)
##
##    Parameter Estimate:  0.26316
##     Estimated Std Err:  0.031652
##               p-value:  <2e-16
##     95% Conf Interval: (0.20112, 0.32519)
## LTMLE estimate

data_ltmle <- heart_disease %>%
  rename(
    baseline_age = age,
    sex = sex_at_birth,
    race = simplified_race,
    education = college_educ,
    income = income_thousands,
    bp1 = blood_pressure,
    treat1 = blood_pressure_medication,
    bmi1 = bmi,
    chol1 = chol,
    bp2 = blood_pressure_2,
    treat2 = blood_pressure_medication_2,
    bmi2 = bmi_2,
    chol2 = chol_2,
    mortality = mortality
  ) %>%
  select(
    baseline_age, sex, race, education, income,       # time-invariant covariates
    bp1, bmi1, chol1,                                  # time 1 covariates (Lnodes)
    treat1,                                            # time 1 treatment (Anode)
    bp2, bmi2, chol2,                                  # time 2 covariates (Lnodes)
    treat2,                                            # time 2 treatment (Anode)
    mortality                                          # outcome (Ynode)
  )

sl_lib_ltmle <- c("SL.glmnet", "SL.xgboost", "SL.gam") #Removed two SL to prevent crash from using too

plan(multicore, workers = parallel::detectCores() - 1)

ltmle_fit <- ltmle(
  data = data_ltmle,
  Anodes = c("treat1", "treat2"),
  Lnodes = c("bp1", "bmi1", "chol1", "bp2", "bmi2", "chol2"),
  Ynodes = "mortality",
  survivalOutcome = FALSE,
  abar = list(c(1, 1), c(0, 0)),
  SL.library = sl_lib_ltmle
)
```

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

## formula for bp2:

## Q.kplus1 ~ baseline_age + sex + race + education + income + bp1 +     bmi1 + chol1 + treat1

## formula for mortality:

## Q.kplus1 ~ baseline_age + sex + race + education + income + bp1 +     bmi1 + chol1 + treat1 + bp2 +

##

## gform not specified, using defaults:

## formula for treat1:
```

```
## treat1 ~ baseline_age + sex + race + education + income + bp1 +     bmi1 + chol1
```

```
## formula for treat2:
```

```
## treat2 ~ baseline_age + sex + race + education + income + bp1 +     bmi1 + chol1 + treat1 + bp2 + bm
```

```
##
```

```
## 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 h(simpleError(msg, call)) :
##   error in evaluating the argument 'x' in selecting a method for function 'drop': non-conformable ar
## 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 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 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: 6 to 35 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
## 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)

## 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 FUN(X[[i]], ...): Coefficients already 0 for all failed algorithm(s)
```

```
# Reporting
summary(ltmle_fit)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = data_ltmle, Anodes = c("treat1", "treat2"), Lnodes = c("bp1",
##     "bmi1", "chol1", "bp2", "bmi2", "chol2"), Ynodes = "mortality",
##     survivalOutcome = FALSE, abar = list(c(1, 1), c(0, 0)), SL.library = sl_lib_ltmle)
##
## Treatment Estimate:
##    Parameter Estimate:  0.19138
##     Estimated Std Err:  0.083084
##               p-value:  0.021251
##     95% Conf Interval: (0.028542, 0.35423)
##
## Control Estimate:
##    Parameter Estimate:  0.56748
##     Estimated Std Err:  0.0058664
##               p-value:  <2e-16
##     95% Conf Interval: (0.55599, 0.57898)
##
## Additive Treatment Effect:
##    Parameter Estimate:  -0.3761
##     Estimated Std Err:  0.083291
##               p-value:  6.3165e-06
##     95% Conf Interval: (-0.53935, -0.21285)
##
## Relative Risk:
##    Parameter Estimate:  0.33725
##   Est Std Err log(RR):  0.43425
##               p-value:  0.012313
##     95% Conf Interval: (0.14399, 0.78992)
##
## Odds Ratio:
##    Parameter Estimate:  0.18039
##   Est Std Err log(OR):  0.5374
##               p-value:  0.0014382
##     95% Conf Interval: (0.062917, 0.51719)
```

The treatment effect estimate for the naive model is larger than the treatment estimate in the model incorporating time varying confounding

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?

In general, we care about confounding in situations where a covariate affects the treatment and the outcome. Time-dependent confounding is particularly an issue when a coviarate is affected itself by prior treatment. Something like blood pressure measured at two different times is a good example of this: Blood pressure in period 2 is impacted by medication exposure in period 1, predicts medication exposure in period 2 and is also a predictor of mortality. On the other hand, while age does vary over time, someone's age in period 1 is perfectly collinear with their age in period 2 so there is no new information obtained in the further observation of their age. In line with this, age is not affected by prior treatment. Therefore we do not need to consider this a time-varying confounder in our LTMLE models.