Project 8 - Jacqueline Brown

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

theme_set(theme_dag())

heart_disease <- read_csv(here('~/git/Computational-Social-Science-Training-Program/Projects/Project 8/
```

```
## 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
# set seed
set.seed(44)

## Train/Test split
# initial_split function from tidymodels/rsample
heart_disease_t1 <- heart_disease %>%
  select(-ends_with("_2"))

## Train/Test split
# initial split
# ----------
heart_split <-
  initial_split(heart_disease_t1 , prop = 3/4) # create initial split (tidymodels)

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

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

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

# Testing
# ----------
test <-
  # Declare the training set with rsample::training()
  testing(heart_split)

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

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

## 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"
```

```
## Train SuperLearner
heart_sl <- mcSuperLearner(Y = y_train,
                           X = x_train,
                           family = binomial(),
                           SL.library = c("SL.glmnet", "SL.mean", "SL.ranger", "SL.knn", "SL.svm"))
```

```
## Loading required namespace: e1071
```

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

```
##
## Call:
## mcSuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = c("SL.glmnet",
##     "SL.mean", "SL.ranger", "SL.knn", "SL.svm"))
##
##
##                     Risk      Coef
## SL.glmnet_All 0.2345146 0.2773597
## SL.mean_All   0.2496041 0.0000000
## SL.ranger_All 0.2303560 0.5596828
## SL.knn_All    0.2756151 0.0000000
## SL.svm_All    0.2362739 0.1629574
```

```
## Discrete winner and superlearner ensemble performance
# predictions
# ----------
preds <-
  predict(heart_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)
```

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

```
# start with y_test
validation <-
  y_test %>%
  # add our predictions - first column of predictions
  bind_cols(preds$pred[,1]) %>%
  # rename columns
  rename(obs = `...1`,       # actual observations
         pred = `...2`) %>% # predicted prob
  # change pred column so that obs above .5 are 1, otherwise 0
  mutate(pred = ifelse(pred >= .5,
                           1,
                           0))
```

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

```
head(validation)
```

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

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  383  209
##          1  860 1048
##
##                Accuracy : 0.5724
##                  95% CI : (0.5527, 0.5919)
##     No Information Rate : 0.5028
##     P-Value [Acc > NIR] : 1.79e-12
##
##                   Kappa : 0.1423
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3081
```

```
##             Specificity : 0.8337
##          Pos Pred Value : 0.6470
##          Neg Pred Value : 0.5493
##              Prevalence : 0.4972
##          Detection Rate : 0.1532
##    Detection Prevalence : 0.2368
##       Balanced Accuracy : 0.5709
##
##        '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)?

Answer: The advantage of SuperLearner is that it increases stability, robustness, and performance. Ensemble modeling allows for more flexibility than selecting the algorithm with the smallest risk score.

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 $P(A|W)$

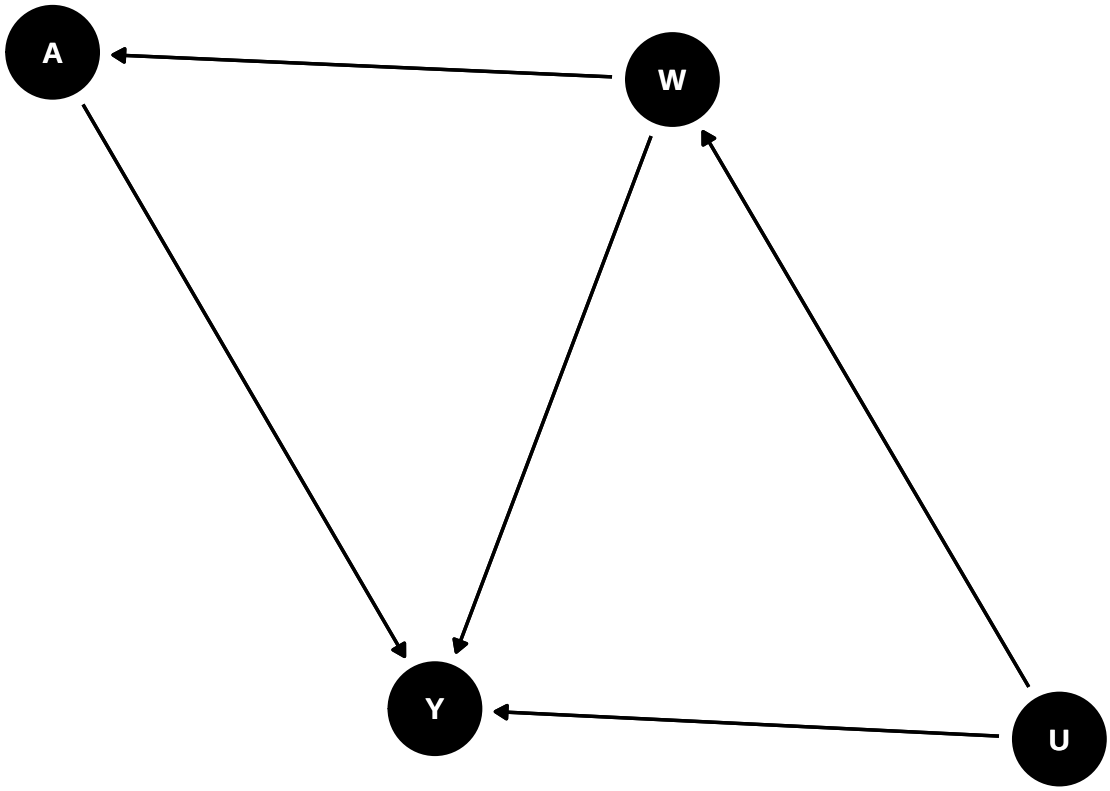
Using ggdag and daggity, draw a directed acylcic graph (DAG) that describes the relationships between the outcome, treatment, and covariates/predictors. Note, if you think there are covariates that are not related to other variables in the dataset, note this by either including them as freestanding nodes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

Answer: In the DAG, Y is the outcome variable, mortality. A is the treatment (taking blood pressure medication), and W are the covariates: demographic characteristics (age, sex, race/ethnicity, household income, and education) and health characteristics (BMI, cholesterol, blood pressure). I did not remove any covariates as they all seem relevant to the outcome.

```
# DAG for TMLE
set.seed(51)

dagify(Y ~ A,
       Y ~ U,
       Y ~ W,
       W ~ U,
       A ~ W,
       exposure = "A",
```

```
      outcome = "Y") %>%
 tidy_dagitty() %>%
# pretty_dag() %>%
 ggdag() +
 geom_dag_edges() +
# geom_dag_node(aes(color = color)) +
 geom_dag_text(col = "white") +
 theme(legend.position = "none")
```




```
 #scale_color_manual(values=c("darkred", "lightgrey", "darkgrey", "navy"))
```

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

```r
#TMLE

# identify number of cores to use
n_cores <- availableCores() - 1 # maybe 2 cores instead?


# plan separate session
plan(multisession,
     workers = n_cores)

# set seed - option will set same seed across multiple cores
set.seed(44, "L'Ecuyer-CMRG")

sl_libs <- c("SL.glmnet", "SL.mean", "SL.ranger", "SL.knn", "SL.svm")


# prep data
#Q <- cv_sl

Y <- heart_disease_t1 %>%
  pull(mortality)

W <- heart_disease_t1 %>%
  select(-mortality) %>%
  select(-blood_pressure_medication)

A <- heart_disease_t1 %>%
  pull(blood_pressure_medication)


# Same outcome and propensity model from DAG.
tmle_fit <-
  tmle::tmle(Y = Y,
           A = A,
           W = W,
           Q.SL.library = sl_libs,
           g.SL.library = sl_libs)
```

```
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
```

```
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in (function (Y, X, newX, family, k = 10, ...)  :
##   SL.knn only available for family = binomial()
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
```

```
tmle_fit
```

```
##  Additive Effect
##    Parameter Estimate:  -0.35425
##    Estimated Variance:  6.5629e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.37013, -0.33838)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.32128
##    Estimated Variance:  0.00014442
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.34483, -0.29772)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.37006
##    Estimated Variance:  5.8296e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.38502, -0.35509)
```

Discussion Questions

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

Answer: A double robust estimator is used to estimate treatment effects in observational data. It is "double robust" because it allows for correct estimation even if either the outcome model or propensity score model were misspecified. The estimator combines the models in a way that allows for one or the other to be misspecified, using a weighted combination of the outcome model and propensity score model. The models are used to weight each other, meaning that if the propensity score model is misspecified (i.e. the matched samples aren't balanced), the outcome model will produce an unbiased estimate of the treatment effect. If the outcome model is misspecified, the double robust estimator produces a robust estimate by using the propensity score model to weight the outcome model. The reason this is so important is that traditional modeling for RCTs makes assumptions about the sample's relationship between outcome, controls, and treatment, but these assumptions can be violated. Double robust estimation allows for misspecification without sacrificing the robustness of the ATE estimators.

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.

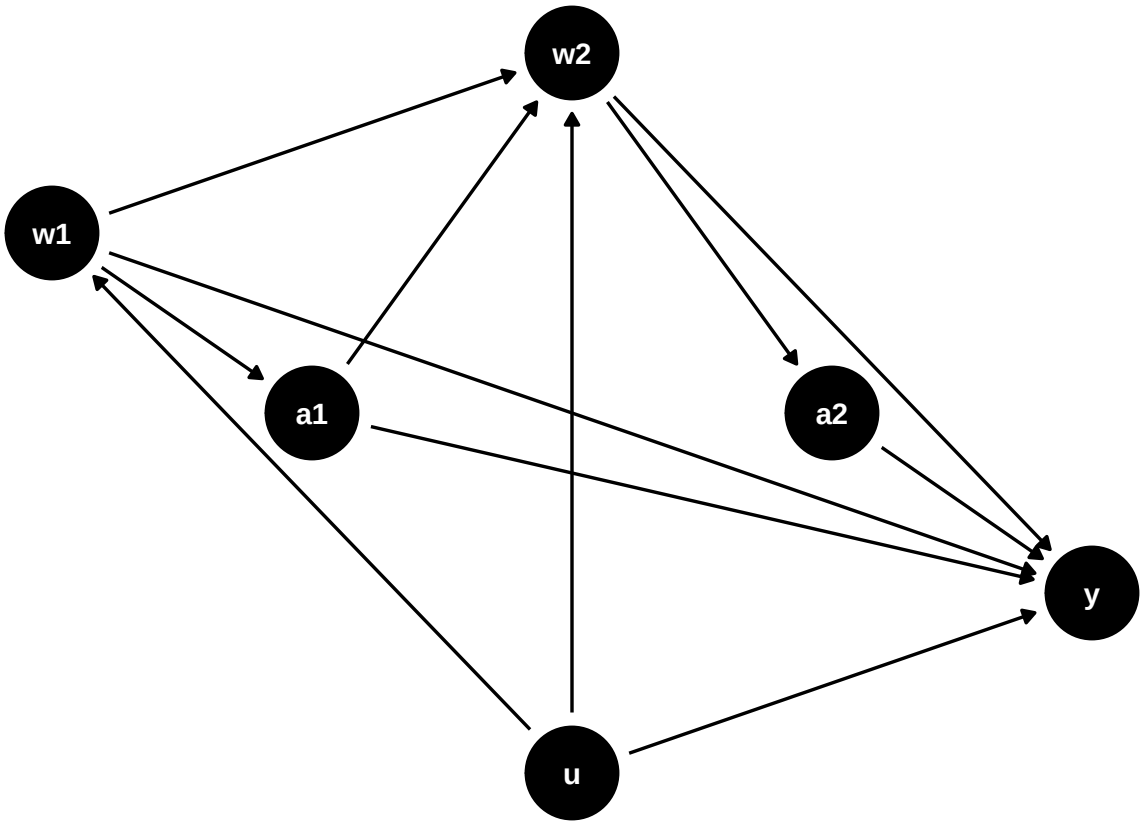
Hint: Check out slide 27 from Maya's lecture, or slides 15-17 from Dave's second slide deck in week 8 on matching.

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
coord_dag <- list(
  x = c(w1 = 0, a1 = 1, w2 = 2, u = 2, a2 = 3, y= 4),
  y = c(w1 = 3, a1 = 2, w2 = 4, u = 0, a2 = 2, y= 1)
)

ltmle_dag <- ggdag::dagify(y ~ a2 + u + w2 + a1 + w1,
                                a2 ~ w2,
                                w2 ~ u + a1 + w1,
                                a1 ~ w1,
                                w1 ~ u,
                           coords = coord_dag)

ggdag::ggdag(ltmle_dag) + theme_void()
```



LTMLE Estimation

Use the `ltmle` package for this section. First fit a "naive model" that **does not** control for the time-dependent confounding. Then run a LTMLE model that does control for any time dependent confounding. Follow the same steps as in the TMLE section. Do you see a difference between the two estimates? **Answer**: Yes, there are considerable differences between the two estimates.

```
## Naive Model (no time-dependent confounding) estimate
sl_libs2 <- c("SL.glmnet", "SL.mean", "SL.svm", "SL.knn")
#removed ranger bc throwing error


Y <- heart_disease %>%
  pull(mortality)

L <- heart_disease %>%
  select(bmi_2, blood_pressure_2, chol_2)

#A1 <- heart_disease %>%
  #pull(blood_pressure_medication)

A2 <- heart_disease %>%
  pull(blood_pressure_medication_2)

W <- heart_disease %>%
```

```r
  select(age, sex_at_birth, simplified_race, college_educ, income_thousands, bmi, blood_pressure, chol)

data <- data.frame(W, L, A2, Y)

## LTMLE estimate
ltmle(data,
      Anodes="A2",
      #Lnodes=NULL,
      Ynodes="Y",
      abar=1,
      SL.library = sl_libs2)
```

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

## formula for Y:

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

##

## gform not specified, using defaults:

## formula for A2:

## A2 ~ age + sex_at_birth + simplified_race + college_educ + income_thousands +     bmi + blood_pressu

##

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
```

```
## Estimate of time to completion: 3 to 11 minutes

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.
```

```
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm  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)

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
```

```
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Call:
## ltmle(data = data, Anodes = "A2", Ynodes = "Y", abar = 1, SL.library = sl_libs2)
##
## TMLE Estimate:  0.5102592
```

```
## LTMLE estimate

sl_libs2 <- c("SL.glmnet", "SL.mean", "SL.svm", "SL.knn")
#removed ranger bc throwing error

Y <- heart_disease %>%
  pull(mortality)

L <- heart_disease %>%
  select(bmi_2, blood_pressure_2, chol_2)

A1 <- heart_disease %>%
  pull(blood_pressure_medication)

A2 <- heart_disease %>%
  pull(blood_pressure_medication_2)

W <- heart_disease %>%
  select(age, sex_at_birth, simplified_race, college_educ, income_thousands, bmi, blood_pressure, chol)

data <- data.frame(W, A1, L, A2, Y)

## LTMLE estimate
ltmle(data,
      Anodes=c("A1", "A2"),
      Lnodes=c("bmi_2",  "blood_pressure_2", "chol_2"),
```

```
      Ynodes="Y",
      abar=c(1, 1),
      SL.library = sl_libs2)
```

```
## 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 Y:

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

##

## gform not specified, using defaults:

## formula for A1:

## A1 ~ age + sex_at_birth + simplified_race + college_educ + income_thousands +     bmi + blood_pressu

## formula for A2:

## A2 ~ age + sex_at_birth + simplified_race + college_educ + income_thousands +     bmi + blood_pressu

##

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
```

```
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

## Estimate of time to completion: 13 to 170 minutes

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.
```

```
## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
```

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm  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)

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!
```

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

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

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class,  :
##   specified nu is infeasible!

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm  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)

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## Warning in svm.default(y = as.factor(Y), x = X, nu = nu, type = type.class, :
## Variable(s) 'Xx.1' constant. Cannot scale data.

## 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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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)

## Warning in FUN(X[[i]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm
##   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]], ...): Variable(s) 'Xx.1' constant. Cannot scale data.

## Error : vector memory exhausted (limit reached?)

## Warning in FUN(X[[i]], ...): Error in algorithm SL.svm  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)

## Call:
## ltmle(data = data, Anodes = c("A1", "A2"), Lnodes = c("bmi_2",
##     "blood_pressure_2", "chol_2"), Ynodes = "Y", abar = c(1,
##     1), SL.library = sl_libs2)
##
## TMLE Estimate:  0.193914
```

Discussion Questions

1. What sorts of time-dependent confounding should we be especially worried about? For instance, would we be concerned about a running variable for age the same way we might be concerned about blood pressure measured at two different times?

Answer: No, we need not be concerned about a running variable for age in the same way we'd be concerned about two different blood pressure measurements. Time-dependent confounding is a concern for LTMLE when estimating an effect of the time-varying treatment on an outcome. Variables that vary over time and are associated with both the outcome and the treatment could bias the estimate. If someone took Asprin at T1, this affects their blood pressure at T2, and since blood pressure is associated with the outcome (heart disease-related death) and the treatment (taking a blood pressure medication), blood pressure is a time-dependent confounder. Age does change over time, but it has a known variance and increases at the same rate for all participants. Blood pressure variance is not predictable, unlike age, which is what makes it a time-dependent confounder. When conducting LTMLE, the main concerns are time-varying confounders, time-dependent selection bias (i.e. dropping out of the study), and time-varying effect modification (changes in severity of disease, etc.).