Project 8 Template

```r
# Add to this package list for additional SL algorithms
gc(); rm(list=ls())
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 474567 25.4    1025657 54.8   660497 35.3
## Vcells 883030  6.8    8388608 64.0  1770414 13.6
```

```r
pacman::p_load(
  tidyverse,
  dplyr,
  ggthemes,
  ltmle,
  tmle,
  SuperLearner,
  tidymodels,
  caret,
  dagitty,
  ggdag,
  here,
  arm,
  xgboost,
  future)

heart_disease <- read_csv('heart_disease_tmle.csv') %>% tibble()
```

```
## 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.library <- c("SL.glm", "SL.glmnet", "SL.bayesglm", "SL.randomForest", "SL.nnet",
                "SL.xgboost", "SL.nnet", "SL.knn", "SL.lda", "SL.ranger")



## Train/Test split
set.seed(123)
heart_split <- initial_split(heart_disease, prop = 3/4)



## Train SuperLearner
heart_train <- training(heart_split)

y_train <- heart_train %>% pull(mortality)
x_train <- heart_train %>%
  dplyr::select(-mortality, -matches(".*_2$"))


heart_test <- testing(heart_split)

y_test <- heart_test %>% pull(mortality)
x_test <- heart_test %>%
  dplyr::select(-mortality, -matches(".*_2$"))



## Risk and Coefficient of each model
superLL <- SuperLearner(Y = y_train,
                        X = x_train,
                        family = binomial(),
                        SL.library = SL.library)
```

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

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

```r
superLL
```

```
##
## Call:
## SuperLearner(Y = y_train, X = x_train, family = binomial(), SL.library = SL.library)
##
##
##
##                           Risk       Coef
## SL.glm_All           0.2357025 0.00000000
## SL.glmnet_All       0.2356664 0.00000000
## SL.bayesglm_All     0.2356999 0.00000000
## SL.randomForest_All 0.2315994 0.05554043
## SL.nnet_All         0.2478574 0.02534190
## SL.xgboost_All      0.2500850 0.00000000
## SL.nnet_All         0.2494901 0.00000000
## SL.knn_All          0.2776735 0.00000000
## SL.lda_All          0.2356699 0.33504426
## SL.ranger_All       0.2312589 0.58407342
```

```
## Discrete winner and superlearner ensemble performance
preds <- predict(superLL,
                 x_test,
                 onlySL = TRUE)

validation <- y_test %>%
  bind_cols(preds$pred[,1]) %>%
  rename(obs = `...1`,
         pred = `...2`) %>%
  mutate(pred = ifelse(pred >= 0.5, 1, 0))
```

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

```
head(validation)
```

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

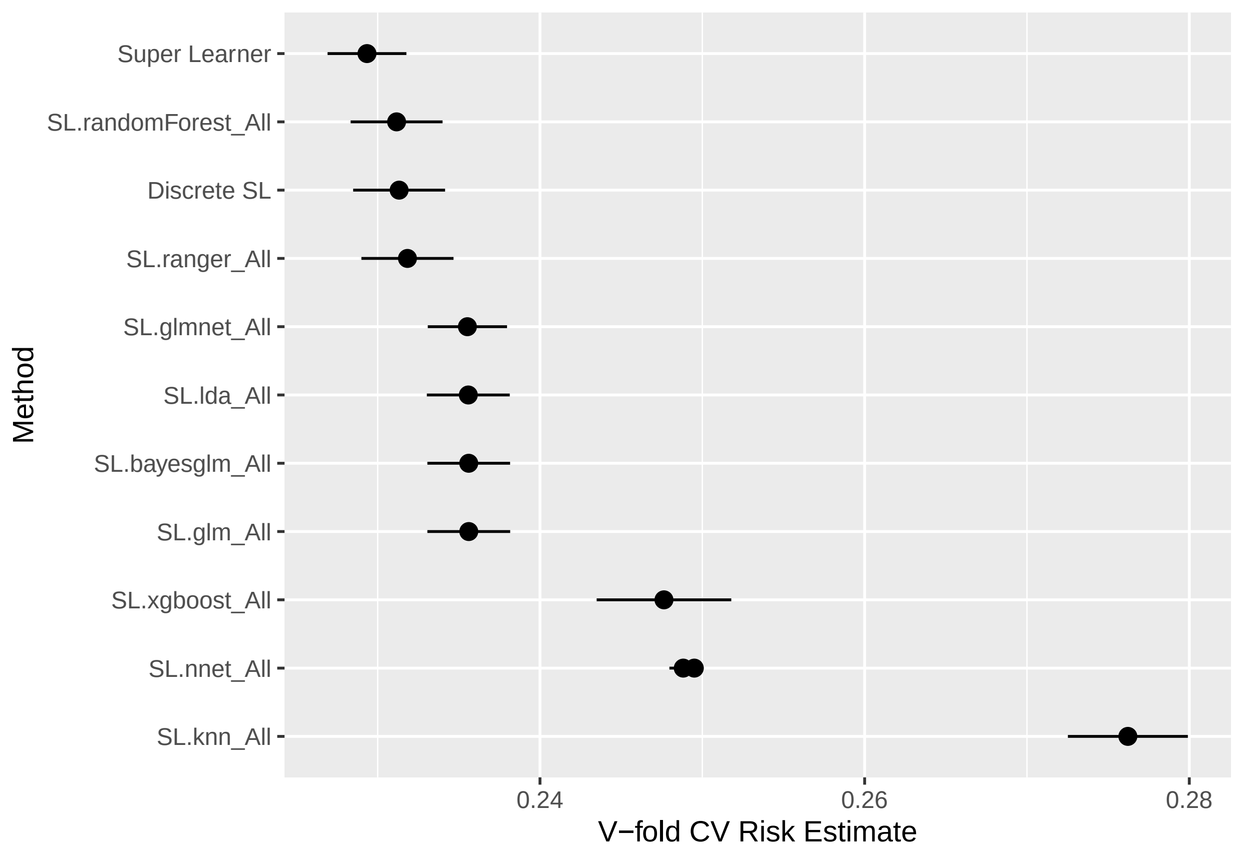
```
# Cross-validation across cores and Plot
cluster = parallel::makeCluster(availableCores() - 1)
parallel::clusterEvalQ(cluster, library(SuperLearner))
```

```
## [[1]]
##  [1] "SuperLearner" "gam"          "foreach"      "splines"      "nnls"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
##
## [[2]]
##  [1] "SuperLearner" "gam"          "foreach"      "splines"      "nnls"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
##
## [[3]]
##  [1] "SuperLearner" "gam"          "foreach"      "splines"      "nnls"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
##
## [[4]]
##  [1] "SuperLearner" "gam"          "foreach"      "splines"      "nnls"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
##
## [[5]]
##  [1] "SuperLearner" "gam"          "foreach"      "splines"      "nnls"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
##
## [[6]]
##  [1] "SuperLearner" "gam"          "foreach"      "splines"      "nnls"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
##
## [[7]]
##  [1] "SuperLearner" "gam"          "foreach"      "splines"      "nnls"
##  [6] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
## [11] "methods"      "base"
```

```r
parallel::clusterSetRNGStream(cluster, 1)


cv_superLL = CV.SuperLearner(Y = y_train,
                             X = x_train,
                             family = binomial(),
                             V = 20,
                             parallel = cluster,
                             SL.library = SL.library)
parallel::stopCluster(cluster)

plot(cv_superLL)
```



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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  395  182
##          1  871 1052
##
##                Accuracy : 0.5788
##                  95% CI : (0.5592, 0.5983)
##     No Information Rate : 0.5064
##     P-Value [Acc > NIR] : 2.287e-13
##
##                   Kappa : 0.1634
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3120
##             Specificity : 0.8525
##          Pos Pred Value : 0.6846
##          Neg Pred Value : 0.5471
##              Prevalence : 0.5064
##          Detection Rate : 0.1580
##    Detection Prevalence : 0.2308
##       Balanced Accuracy : 0.5823
##
##        'Positive' Class : 0
##
```

```
TP <- CM$table[2,2]
TN <- CM$table[1,1]
FP <- CM$table[2,1]
FN <- CM$table[1,2]

print(paste("accuracy: ", (TP+TN)/(TP+TN+FP+FN)))
```

```
## [1] "accuracy:  0.5788"
```

```
print(paste("recall: ", TP/(TP+FN)))
```

```
## [1] "recall:  0.852512155591572"
```

```
print(paste("precision: ", TP/(TP+FP)))
```

```
## [1] "precision:  0.547061882475299"
```

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 SuperLearner ensemble is preferred over a single "best" model due to its "meta-model" which uses out-of-sample predictions from all participating models. The SuperLearner integrates the strengths of various models, using a strategic combination that allows it to consistently outperform any individual model in the ensemble. Additionally, it enhances accuracy by refining predictions based on actual performance, reducing biases and errors often overlooked in initial evaluations.

Targeted Maximum Likelihood Estimation

Causal Diagram

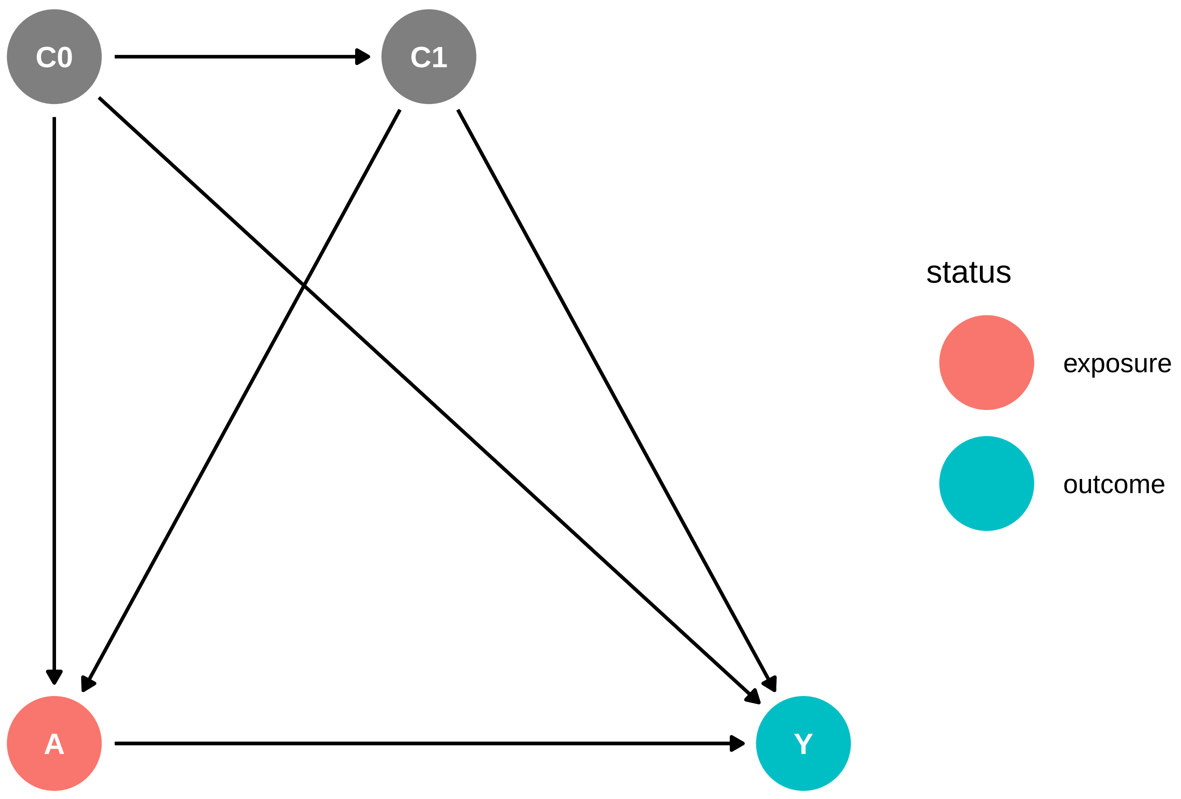
TMLE requires estimating two models:

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

dagify(Y ~ A + C0 + C1,
       A ~ C0 + C1,
       C1 ~ C0,
       exposure = "A",
       outcome = "Y",
       coords = list(x = c(A=1, C0=1, C1=2, Y=3),
                     y = c(A=1, C0=1.5, C1=1.5, Y=1))) %>%
  ggdag_status() +
  geom_dag_edges() +
  theme_dag()
```



```r
data.frame(
  Variable = c("A", "C0", "C1", "Y"),
  Description = c("BP Med (treatment)",
                  "Age, Sex, Race, Income, College Ed",
                  "BMI, Chol, BP",
                  "Mortality (outcome)"))
```

```
##   Variable                        Description
## 1        A                  BP Med (treatment)
## 2       C0 Age, Sex, Race, Income, College Ed
## 3       C1                       BMI, Chol, BP
## 4        Y                 Mortality (outcome)
```

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

NOTE. In this scenario, combining C0 and C1 involves the assumption that each variable does not exert an independent effect, but rather that their effects can be expressed through a single combined variable. By treating C0 and C1 as covariates on the same level, the model does not account for the influence of C0 on C1 nor does it differentiate between their direct and indirect effects on A or Y. When C0 and C1 are grouped together, it is assumed that these two variables are closely interrelated, and the impact of their relationship can be effectively captured by a single composite variable. Nevertheless, integrating C0 and C1 allows the model to account for both their direct and indirect impacts on A and Y. The simpler model would be "Y ~ A + C, A ~ C, C = C0 + C1"

```r
# Outcome
Y <- heart_disease %>%
  pull(mortality)

A <- heart_disease %>%
  pull(blood_pressure_medication)

W <- heart_disease %>%
  dplyr::select(-mortality, -blood_pressure_medication, -matches(".*_2$"))

#SL.library <- c("SL.glm", "SL.bayesglm", "SL.glmnet", "SL.randomForest", "SL.xgboost", "SL.nnet", "SL.
SL.library2 <- c('SL.glmnet', 'SL.ranger', 'SL.glm')
tmle_fit <- tmle::tmle(Y = Y,
                       A = A,
                       W = W,
                       Q.SL.library = SL.library2,
                       g.SL.library = SL.library2)
```

```
## Growing trees.. Progress: 66%. Estimated remaining time: 15 seconds.
## Growing trees.. Progress: 94%. Estimated remaining time: 1 seconds.
```

```r
tmle_fit
```

```
##  Additive Effect
##    Parameter Estimate:  -0.3542
##    Estimated Variance:  6.3284e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.36979, -0.3386)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.31987
##    Estimated Variance:  0.00014504
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.34348, -0.29627)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.37182
##    Estimated Variance:  5.5899e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.38647, -0.35717)
```

```r
summary(tmle_fit)
```

```
##  Initial estimation of Q
##   Procedure: cv-SuperLearner, ensemble
##   Model:
##       Y ~  SL.glmnet_All + SL.ranger_All + SL.glm_All
##
##   Coefficients:
##     SL.glmnet_All    0.2932814
##     SL.ranger_All    0.7067186
##        SL.glm_All    0
##
##   Cross-validated R squared :  0.0834
##
##  Estimation of g (treatment mechanism)
##   Procedure: SuperLearner, ensemble
##   Model:
##       A ~  SL.glmnet_All + SL.ranger_All + SL.glm_All
##
##   Coefficients:
##     SL.glmnet_All    0.7272909
##     SL.ranger_All    0.1747352
##        SL.glm_All    0.09797395
##
##  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/ATE: (0.0054, 0.9946)
##
##  Additive Effect
##    Parameter Estimate:  -0.3542
##    Estimated Variance:  6.3284e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.36979, -0.3386)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  -0.31987
##    Estimated Variance:  0.00014504
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.34348, -0.29627)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  -0.37182
##    Estimated Variance:  5.5899e-05
##               p-value:  <2e-16
##     95% Conf Interval:  (-0.38647, -0.35717)
```

The average treatment effect is -0.35373, the average treatment effect among the treated is -0.32067, and the average treatment effect among the control group is -0.37379.

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: "double robust" estimator is a statistical estimation method that remains consistent if either of two models — the outcome model or the propensity score model — is correctly specified. If the propensity score model is correct, it achieves an accurate distribution of treatment assignment, which allows balanced covariates between the treatment groups, which is achieved in a randomized study. Thus, even if the outcome model is wrong, the adjustment for confounders via the correctly specified propensity score leads to an unbiased estimation of the treatment effect. On the other hand, if the outcome model is correct, it accurately captures the relationship between the treatment, covariates, and the outcome. So, the outcome model itself provides a correct estimate of the expected outcome conditional on treatment and covariates.

LTMLE Estimation

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

Causal Diagram

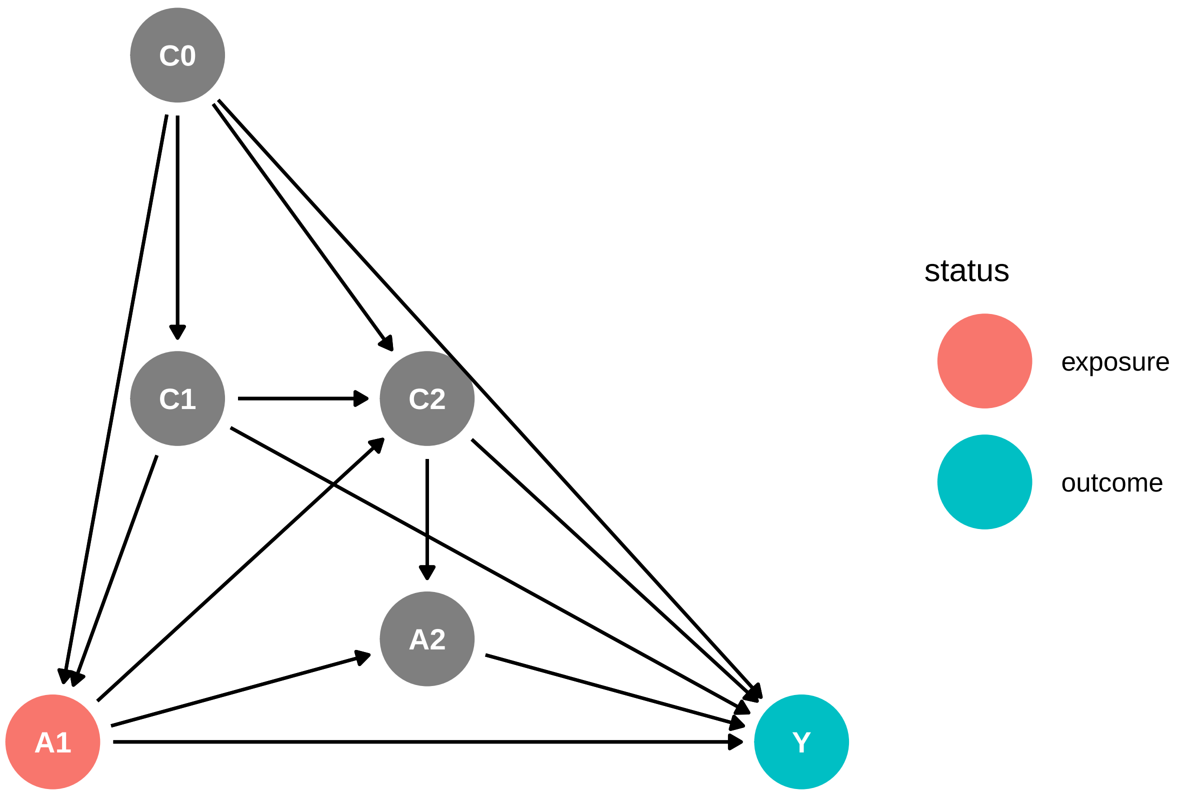
Update your causal diagram to incorporate this new information. **Note**: If your groups divides up sections and someone is working on LTMLE separately from TMLE then just draw a causal diagram even if it does not match the one you specified above.

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

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

```r
# DAG for TMLE

dagify(Y ~ A1 + A2 + C0 + C1 + C2,
       A2 ~ A1 + C2,
       A1 ~ C0 +C1,
       C2 ~ A1 + C0 + C1,
       C1 ~ C0,
       exposure = "A1",
       outcome = "Y",
       coords = list(x = c(A1=1, C0=1.5, C1=1.5, C2=2.5, A2=2.5, Y=4),
                     y = c(A1=1, C0=3, C1=2, C2=2, A2=1.3, Y=1))) %>%
  ggdag_status() +
  geom_dag_edges() +
  theme_dag()
```



```r
data.frame(
  Variable = c("A1", "A2", "C0", "C1", "C2", "Y"),
  Description = c("BP Med (treatment)",
                  "BP Med_2 (treatment)",
                  "Age, Sex, Race, Income, College Ed",
                  "BMI, Chol, BP",
                  "BMI_2, Chol_2, BP_2",
                  "Mortality (outcome)"))
```

```
##   Variable                        Description
## 1      A1                 BP Med (treatment)
## 2      A2               BP Med_2 (treatment)
## 3      C0 Age, Sex, Race, Income, College Ed
## 4      C1                      BMI, Chol, BP
## 5      C2                BMI_2, Chol_2, BP_2
## 6       Y                Mortality (outcome)
```

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

A1 <- heart_disease %>% pull(blood_pressure_medication)
A2 <- heart_disease %>% pull(blood_pressure_medication_2)

C0_1 <- heart_disease %>% pull(age)
C0_2 <- heart_disease %>% pull(sex_at_birth)
C0_3 <- heart_disease %>% pull(simplified_race)
C0_4 <- heart_disease %>% pull(college_educ)
C0_5 <- heart_disease %>% pull(income_thousands)

C1_1 <- heart_disease %>% pull(bmi)
C1_2 <- heart_disease %>% pull(blood_pressure)
C1_3 <- heart_disease %>% pull(chol)

C2_1 <- heart_disease %>% pull(bmi_2)
C2_2 <- heart_disease %>% pull(blood_pressure_2)
C2_3 <- heart_disease %>% pull(chol_2)

Y <- heart_disease %>% pull(mortality)

data1 <- data.frame(C0_1,C0_2,C0_3,C0_4,C0_5,A1,A2,Y)
data2 <- data.frame(C0_1,C0_2,C0_3,C0_4,C0_5,C1_1,C1_2,C1_3,C2_1,C2_2,C2_3,A1,A2,Y)

result1_1 <- glm(Y ~ A1+A2, data = data1)

result1_2 <- ltmle(data1,
                   Anodes = c("A1","A2"),
                   Lnodes = NULL,
                   Ynodes = "Y",
                   abar = c(1, 1),
                   SL.library = SL.library2
                   )
```

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

## formula for Y:

## Q.kplus1 ~ C0_1 + C0_2 + C0_3 + C0_4 + C0_5 + A1 + A2
```

```
##

## gform not specified, using defaults:

## formula for A1:

## A1 ~ C0_1 + C0_2 + C0_3 + C0_4 + C0_5

## formula for A2:

## A2 ~ C0_1 + C0_2 + C0_3 + C0_4 + C0_5 + A1

##

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

```
result1_1$coefficients[1]
```

```
## (Intercept)
##   0.5657497
```

```
result2 <- ltmle(data2,
                 Anodes = c("A1", "A2"),
                 Lnodes = c("C1_1", "C1_2", "C1_3","C2_1", "C2_2", "C2_3"),
                 Ynodes = "Y",
                 gform = c("A1 ~ C1_1 + C1_2 + C1_3",
                            "A2 ~ A1 + C2_1 + C2_2 + C2_3"),
                 abar = c(1, 1),
                 SL.library = SL.library2
                 )
```

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

## formula for Y:

## Q.kplus1 ~ C0_1 + C0_2 + C0_3 + C0_4 + C0_5 + C1_1 + C1_2 + C1_3 +     C2_1 + C2_2 + C2_3 + A1 + A2

##

## Estimate of time to completion: 4 to 17 minutes
```

```
dif_val_1 <- round(result1_1$coefficients[2] - result2$estimates[1], 4)
dif_val_2 <- round(result1_2$estimates[1] - result2$estimates[1], 4)

print(paste("Difference in Naive and LTMLE estimate :",dif_val_1))
```

```
## [1] "Difference in Naive and LTMLE estimate : -0.7086"
```

```r
print(paste("Difference in TMLE estimates:", dif_val_2))
```

```
## [1] "Difference in TMLE estimates: -0.1308"
```

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: We need to be worried about time-dependent confounding variables when they are very sensitive to latent conditions. For example, while age influences treatment decisions and outcomes, it remains unaffected by other variables, making its control relatively straightforward without introducing significant bias. On the other hand, blood pressure is a complex time-dependent confounder, influenced by and influencing latent variables, treatments, and outcomes.