Applied Metrics Empirical P-Set 2, Fall 2023

2023-11-29

Contents

0.0.1 Load data

```
# Clear all
rm(list = ls())

# load and require packages
pacman::p_load(foreign, tidyverse, formatR, here, randomForest,
    boot, MASS, ivmte, aod)

# Load data
data <- read.csv(here("data/andres_cleaned.csv"))
data$X <- NULL

set.seed(1234)  # For reproducibility
```

0.0.2 Question 1a - Relate structural equation and probit regression

Question: Suppose we specify the potential outcome equations as $Y_i(1) = X_i'\theta_1 + \epsilon_{1i}$ and $Y_i(0) = X_i'\theta_0 + \epsilon_{0i}$, and the structural selection equation as $D_i = 1\{u(X_i, Z_i) \geq V_i\}$. where $u(X_i, Z_i) - V_i$ is the latent utility derived from additional children. How do you relate the structural selection equation and the probit regression of (1)? Explain your answer.

Answer: The structural selection equation describes the switch from $D = 0$ to $D = 1$ in terms of utility. In other words, the structural equation implies that the family will have three or more children if, given their observables X and instrument Z, the family's utility of doing so is higher than some level V_i. The propensity score quantifies this probability, capturing the likelihood, given your observables and instrument value, of having three of more children ($D = 1$).

0.0.3 Question 1b - Assumptions for MTE

Clarify a set of assumptions that the marginal treatment effect $M(x, u)$, $u \in (0, 1)$ can be identified by the derivative of the OLS regression equation WRT the propensity score:

$$MTE(x, u) = \frac{\partial}{\partial p}\bigg|_{p=u} (x'\beta_0 + px'\beta_1 + \kappa_1 p + \kappa_2 p^2 + \kappa_3 p^3)$$

The assumptions are that (1) the instrument is independent $(U_1, U_0, V) \perp Z|X$, (2) that $\mu_D(Z, X)$ has a nondegenerate distribution given X, (3) that scalar V is continuously distributed, (4) $E(|Y(1)|)$ and $E(|Y(0)|)$ are finite, and (5) $0 < P(D = 1|X) < 1$.

0.0.4 Question 1c - Estimate propensity score

Estimate the propensity score as instructed above, and assess whether the coefficient of the instrumental variable in the probit regression is significantly different from zero or not.

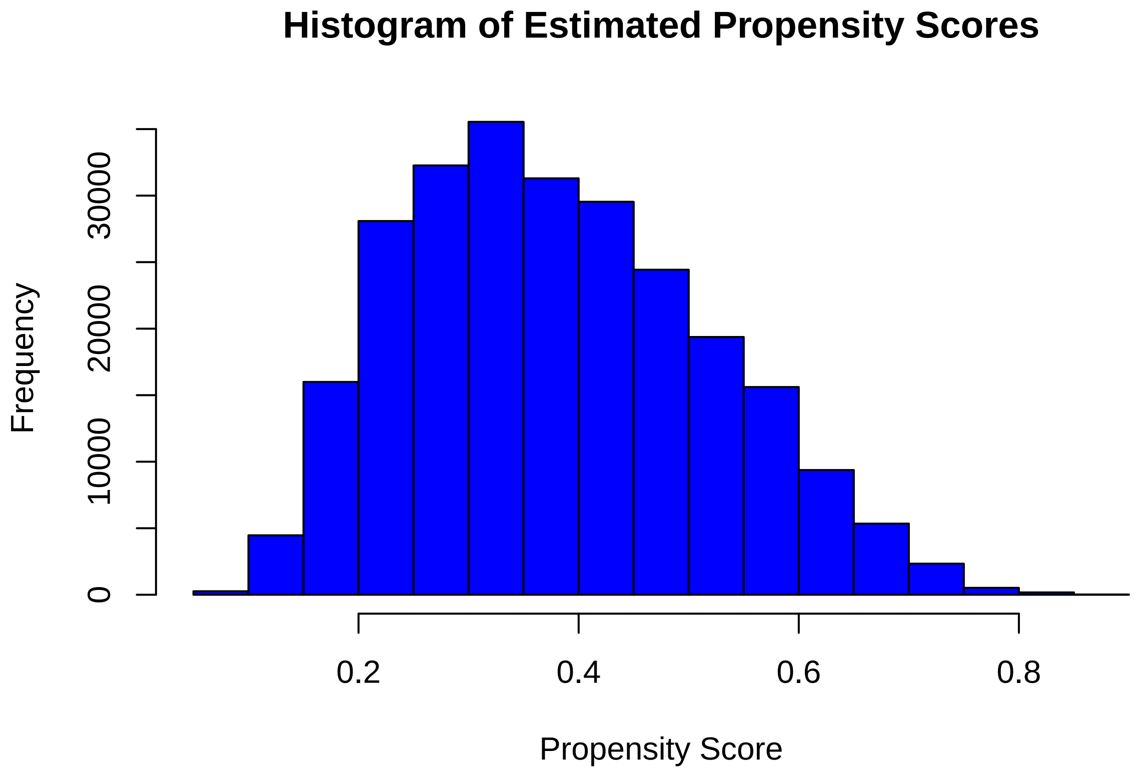
Yes, we see that the z-value for the instrument is extremely large at 37, and very significant. So we have confidence that the instrument is predicting treatment.

```
# Fit probit
probit_model <- glm(d ~ blackm + hispm + othracem + agem1 + agefstm +
    boy1st + z, data = data, family = binomial(link = "probit"))

# Predict propensity scores
data$prop_score <- predict(probit_model, type = "response")

summary(probit_model)
```

```
##
## Call:
## glm(formula = d ~ blackm + hispm + othracem + agem1 + agefstm +
##     boy1st + z, family = binomial(link = "probit"), data = data)
##
## Coefficients:
##               Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -0.3878595  0.0261706  -14.820  < 2e-16 ***
## blackmTRUE   0.1712398  0.0115636   14.808  < 2e-16 ***
## hispmTRUE    0.4462240  0.0159873   27.911  < 2e-16 ***
## othracem     0.1631770  0.0151467   10.773  < 2e-16 ***
## agem1        0.0822504  0.0008432   97.540  < 2e-16 ***
## agefstm     -0.1220241  0.0010082 -121.031  < 2e-16 ***
## boy1stTRUE  -0.0308048  0.0051877   -5.938 2.88e-09 ***
## zTRUE        0.1911553  0.0051907   36.827  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 338352  on 254651  degrees of freedom
## Residual deviance: 317827  on 254644  degrees of freedom
## AIC: 317843
##
## Number of Fisher Scoring iterations: 4
```

0.0.5 Question 1d - Plot histogram of estimated propensity scores

```
hist(data$prop_score, main = "Histogram of Estimated Propensity Scores",
    xlab = "Propensity Score", ylab = "Frequency", col = "blue",
    border = "black")
```



```r
summary(probit_model)
```

```
##
## Call:
## glm(formula = d ~ blackm + hispm + othracem + agem1 + agefstm +
##     boy1st + z, family = binomial(link = "probit"), data = data)
##
## Coefficients:
##               Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -0.3878595  0.0261706  -14.820  < 2e-16 ***
## blackmTRUE   0.1712398  0.0115636   14.808  < 2e-16 ***
## hispmTRUE    0.4462240  0.0159873   27.911  < 2e-16 ***
## othracem     0.1631770  0.0151467   10.773  < 2e-16 ***
## agem1        0.0822504  0.0008432   97.540  < 2e-16 ***
## agefstm     -0.1220241  0.0010082 -121.031  < 2e-16 ***
## boy1stTRUE  -0.0308048  0.0051877   -5.938 2.88e-09 ***
## zTRUE        0.1911553  0.0051907   36.827  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 338352  on 254651  degrees of freedom
## Residual deviance: 317827  on 254644  degrees of freedom
## AIC: 317843
```

```
##
## Number of Fisher Scoring iterations: 4
```

0.0.6 Question 1e - Estimate by OLS, bootstrap standard errors

Report the OLS estimates of the coefficients in the regression equation of (2) and their bootstrap standard errors with 1000 bootstrap replications.

```r
# Fit OLS
ols <- lm(y ~ blackm + hispm + othracem + agem1 + agefstm + boy1st +
    prop_score * (blackm + hispm + othracem + agem1 + agefstm +
        boy1st) + prop_score + I(prop_score^2) + I(prop_score^3),
    data = data)

summary(ols)
```

```
##
## Call:
## lm(formula = y ~ blackm + hispm + othracem + agem1 + agefstm +
##     boy1st + prop_score * (blackm + hispm + othracem + agem1 +
##     agefstm + boy1st) + prop_score + I(prop_score^2) + I(prop_score^3),
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.9609  -0.1833   0.1303   0.4026   2.0208
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           8.600058   0.107316  80.138  < 2e-16 ***
## blackmTRUE           -0.134467   0.034157  -3.937 8.26e-05 ***
## hispmTRUE            -0.516547   0.062459  -8.270  < 2e-16 ***
## othracem             -0.143154   0.037967  -3.771 0.000163 ***
## agem1                 0.068909   0.006096  11.304  < 2e-16 ***
## agefstm              -0.003509   0.008411  -0.417 0.676503
## boy1stTRUE            0.004896   0.012190   0.402 0.687931
## prop_score            1.129845   0.537263   2.103 0.035470 *
## I(prop_score^2)      -1.591842   0.926932  -1.717 0.085921 .
## I(prop_score^3)       1.866082   0.584859   3.191 0.001420 **
## blackmTRUE:prop_score -0.015681   0.073780  -0.213 0.831685
## hispmTRUE:prop_score   0.293836   0.123387   2.381 0.017247 *
## othracem:prop_score   -0.115379   0.091534  -1.261 0.207488
## agem1:prop_score      -0.071998   0.015154  -4.751 2.02e-06 ***
## agefstm:prop_score     0.063574   0.021044   3.021 0.002520 **
## boy1stTRUE:prop_score  0.001898   0.030160   0.063 0.949829
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.026 on 254636 degrees of freedom
## Multiple R-squared:  0.03367,    Adjusted R-squared:  0.03362
## F-statistic: 591.5 on 15 and 254636 DF,  p-value: < 2.2e-16
```

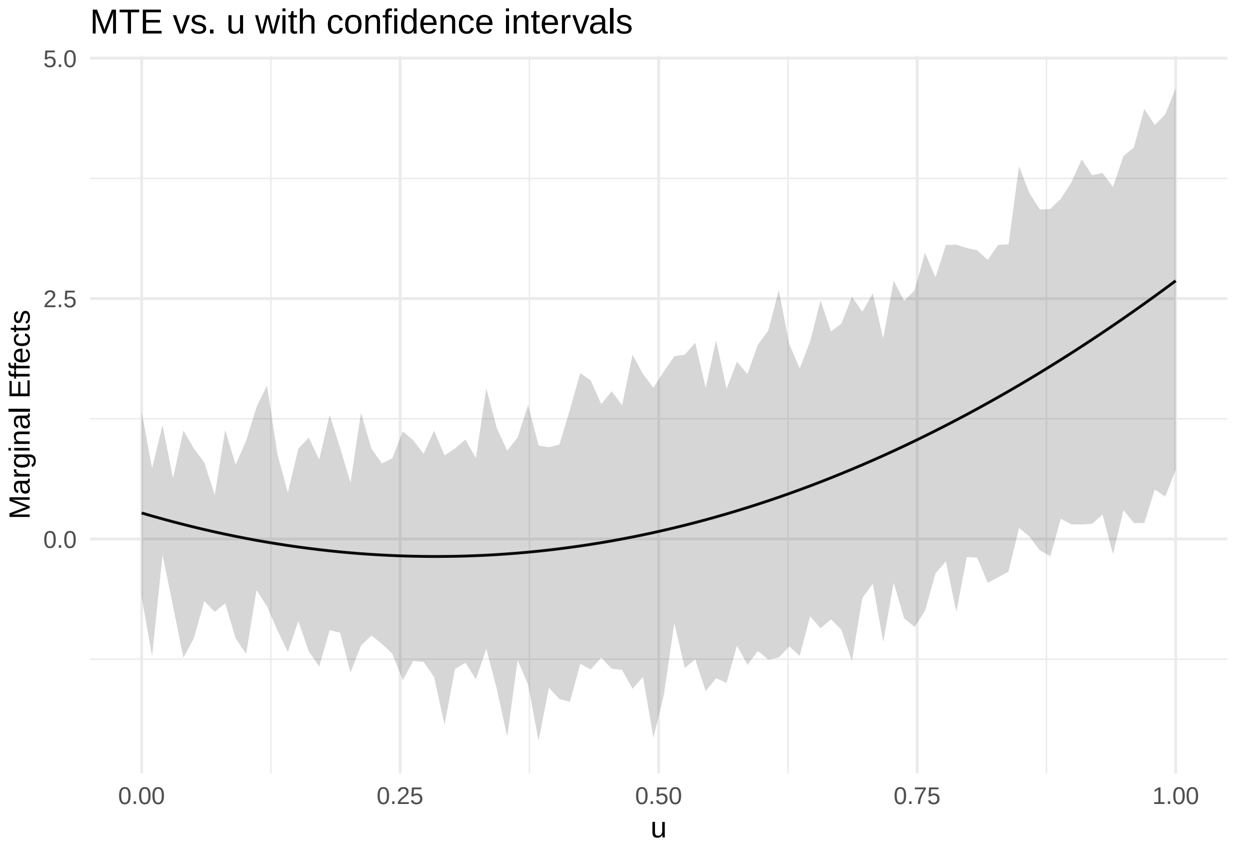
```r
# Function to fit model on bootstrap sample
boot_fn <- function(data, index) {
    fit <- lm(y ~ blackm + hispm + othracem + agem1 + agefstm +
        boy1st + prop_score * (blackm + hispm + othracem + agem1 +
        agefstm + boy1st) + prop_score + I(prop_score^2) + I(prop_score^3),
        data = data, subset = index)
    return(coef(fit))
}

# Perform bootstrap
boot_results <- boot(data, boot_fn, R = 1000)

# Calculate standard errors
boot_se <- apply(boot_results$t, 2, sd)
boot_se
```

```
##  [1] 0.109960720 0.037489230 0.071993780 0.051808754 0.006021527 0.008181801
##  [7] 0.011958861 0.565283406 1.015039125 0.660385117 0.080876546 0.135673840
## [13] 0.125215546 0.015254902 0.020825124 0.030179818
```

0.0.7 Question 1f - Calculate MTE and their CI

Plot the MTE estimates (as a function of u [0,1]) and pointwise confidence intervals, holding X constant at its sample mean values. The confidence intervals are constructed using the bootstrap results.

```r
# Step 1: Calculate the means of the covariates
covariate_means <- colMeans(data[, c("blackm", "hispm", "othracem",
    "agem1", "agefstm", "boy1st")], na.rm = TRUE)

# Calculate the first term of the derivative (x prime times
# beta1)
beta1 <- coef(ols)[11:16]
xprime <- matrix(covariate_means, nrow = 1, ncol = 6)  #transpose to make 1x6
beta1 <- matrix(beta1, nrow = 6, ncol = 1)  #transpose to make 6x1
xbeta1 <- xprime %*% beta1

kappa1 <- coef(ols)[8]  #prop_score coef
kappa2 <- coef(ols)[9]  #prop_score^2 coef
kappa3 <- coef(ols)[10]  #prop_score^3 coef

# define u
u <- seq(0, 1, length.out = 100)

marginal_effects <- as.vector(xbeta1) + as.vector(kappa1) + (2 *
    as.vector(kappa2) * u) + (3 * as.vector(kappa3) * (u)^2)

# Create a new data frame for plotting
plot_data <- data.frame(xvar = u, marginal_effects = marginal_effects)

# Use ggplot to plot MTEs
plot <- ggplot(plot_data, aes(x = u, y = marginal_effects)) +
    geom_point() + theme_minimal() + labs(x = "u", y = "Marginal Effects",
    title = "Scatterplot of Marginal Effects vs. Unobserved Heterogeneity u")

kappa_1_boot <- boot_results$t[, 8]
kappa_2_boot <- boot_results$t[, 9]
kappa_3_boot <- boot_results$t[, 10]
beta_1_boot <- boot_results$t[, 11:16]
xbeta1_boot <- xprime %*% t(beta_1_boot)

marginal_effects_matrix <- as.vector(xbeta1_boot) + as.vector(kappa_1_boot) +
    (2 * u %*% t(kappa_2_boot)) + (3 * u^2 %*% t(kappa_3_boot))

MTE_lb <- apply(marginal_effects_matrix, 1, function(x) quantile(x,
    probs = 0.025))
MTE_ub <- apply(marginal_effects_matrix, 1, function(x) quantile(x,
    probs = 0.975))

# Create a data frame for ggplot
plot_data <- data.frame(u, marginal_effects, MTE_lb, MTE_ub)

# Plot using ggplot
mte_plot <- ggplot(plot_data, aes(x = u, y = marginal_effects)) +
    geom_line() + geom_ribbon(aes(ymin = MTE_lb, ymax = MTE_ub),
    alpha = 0.2) + labs(x = "u", y = "Marginal Effects", title = "MTE vs. u with confidence intervals")
    theme_minimal()

mte_plot
```



0.0.8 Question 1g - Interpret

Question: Based on the presented estimates, interpret heterogeneity of the marginal treatment effects. What type of parent has a larger causal effect than others? In your answer, be explicit about how to interpret u.

Answer: u is the unobservable latent variable in the selection equation. That is, u captures the unobservable propensity to take up treatment. The higher the u, the less likely the person is to have had more than two kids without having had twins.

If MTE is increasing in u, then people who are less likely to choose treatment experience larger treatments effects. In this case, MTE is increasing in u, implying that mothers who are already less likely to have a third child on their own (without the presence of twins) experience more impacts on their income.

0.0.9 Question 1h - Based on the estimation results of part (b), test heterogeneity of MTE in u.

```r
wald_heter <- wald.test(b = coef(ols), Sigma = vcov(ols), Terms = 15:16)
wald_heter
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 9.3, df = 2, P(> X2) = 0.0098
```

Answer: Since non-zero polynomial terms in the MTE equation are what could give a non-linear shape to the MTE function, we can test whether the MTEs are heterogeneous by testing whether κ_2 and κ_3 are equal to zero. The Wald test above rejects the Null that MTEs are homogeneous.

0.0.10 Question 1i - Estimate ATE

Question: Under the specifications of (2) and MTE identification by (5), explain how you identify and estimate the average treatment effect. In your answer, address any concerns about credibility on the ATE estimate.

Answer: I interpret this question as asking me to explain how to compute the ATE, not to actually do it. I would compute the ATE as the average of all the MTEs, integrated over the propensity score: $ATE = \int_p \text{MTE} dp$. There are concerns about the credibility of the ATE because of the heterogeneity of the MTEs – since the ATE is just the average, only reporting the ATE masks the variation for different subgroups.

0.0.11 Question 1j - Non-parametric function of propensity score $K(p)$

Question: To assess robustness of the results relying on the parametric form in 2, consider estimating MTE using partial linear regression: $Y_i = X_i'\beta_0 + \widehat{p}_i X_i'\beta_1 + K(p)$, where $K(p)$ is a nonparamtetric function of the propensity score. Maintain the propensity score estimates as before, estimate MTE by running a partial linear regression on (6). In your answer, explain how you implement the partial linear regression and how you choose the smoothing parameters (bandwith).

Answer:

0.1 Question 2

0.1.1 Question 2a

```r
# Worked with Raquel Badillo-Salas

set.seed(1234)

# Set the number of variables and observations
n_vars <- 20
n_obs <- 500

# Create the variance matrix sigma
sigma <- matrix(0, nrow = n_vars, ncol = n_vars)
for (i in 1:n_vars) {
    for (j in 1:n_vars) {
        sigma[i, j] <- 0.8^abs(j - i)
    }
}

# Ensure diagonal entries are 1
diag(sigma) <- 1

# Generate the data
data2 <- as.data.frame(mvrnorm(n = n_obs, mu = rep(0, n_vars),
    Sigma = sigma))  # Need MASS package
```

```
####################################

# Set up
a0 <- 1
a1 <- 0.25
b0 <- 1
b1 <- 0.25
tau <- 0.5

# Define m and V
m <- function(X1, X3) {
    return(a0 * X1 + a1 * (exp(X3)/(1 + exp(X3))))
}

V_fun <- function(X1, X3) {
    zeta <- rnorm(500, 0, 1)
    return(m(X1, X3) + zeta)
}

V <- V_fun(data2$V1, data2$V3)
data2$D <- ifelse(V > 0, 1, 0)

# Define g(X,D) and Y
g_fun <- function(D, X1, X3) {
    return(tau * D + b0 * (exp(X1)/(1 + exp(X1))) + b1 * X3)
}

Y_fun <- function(X1, X3, D) {
    epsilon <- rnorm(500, 0, 1)
    return(g_fun(D, X1, X3) + epsilon)
}

data2$Y <- Y_fun(data2$V1, data2$V3, data2$D)

####################################

independent_vars <- paste0("V", 1:20, collapse = " + ")
formula <- as.formula(paste("D ~ ", independent_vars))

# Fit probit
probit_model <- glm(formula, data = data2, family = binomial(link = "probit"))

# Predict propensity scores
data2$prop_score <- predict(probit_model, type = "response")

summary(probit_model)
```

```
##
## Call:
## glm(formula = formula, family = binomial(link = "probit"), data = data2)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.141706   0.064261   2.205   0.0274 *
## V1           0.741889   0.116708   6.357 2.06e-10 ***
## V2           0.048855   0.140577   0.348   0.7282
## V3           0.131312   0.146360   0.897   0.3696
## V4          -0.021512   0.139910  -0.154   0.8778
## V5          -0.007132   0.133712  -0.053   0.9575
## V6          -0.184478   0.137450  -1.342   0.1795
## V7           0.282537   0.141918   1.991   0.0465 *
## V8          -0.015244   0.132922  -0.115   0.9087
## V9          -0.005232   0.139040  -0.038   0.9700
## V10         -0.124826   0.141523  -0.882   0.3778
## V11          0.024793   0.139823   0.177   0.8593
## V12         -0.071307   0.141355  -0.504   0.6139
## V13         -0.051754   0.139740  -0.370   0.7111
## V14          0.259755   0.140511   1.849   0.0645 .
## V15         -0.006519   0.140353  -0.046   0.9630
## V16         -0.138994   0.144635  -0.961   0.3366
## V17          0.133267   0.139894   0.953   0.3408
## V18         -0.042113   0.137855  -0.305   0.7600
## V19         -0.078995   0.136308  -0.580   0.5622
## V20         -0.014995   0.097847  -0.153   0.8782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 686.86  on 499  degrees of freedom
## Residual deviance: 534.67  on 479  degrees of freedom
## AIC: 576.67
##
## Number of Fisher Scoring iterations: 5
```

```
data2$w_ate <- (data2$D * data2$Y/(data2[["prop_score"]])) -
    ((1 - data2$D) * data2$Y/(1 - data2[["prop_score"]]))

ate_probit <- mean(data2$w_ate)
ate_probit
```

```
## [1] 0.4618182
```

0.1.2 Question 2b - Estimate ATE using Random Forest.

```
data2 <- subset(data2, select = c(-w_ate, -prop_score))

# Fit Random Forest models Replace 'Y ~ .' with 'Y ~
# [covariates]' to specify covariates
rf_treatment <- randomForest(Y ~ ., data = data2[data2$D == 1,
    ])
rf_control <- randomForest(Y ~ ., data = data2[data2$D == 0,
    ])
```

```r
# Predict outcomes using the fitted models
rf_pred_treatment <- predict(rf_treatment, data2)
rf_pred_control <- predict(rf_control, data2)

# Estimate Average Treatment Effect (ATE)
ate_rf <- mean(rf_pred_treatment) - mean(rf_pred_control)

# Output the ATE
ate_rf
```

```
## [1] 0.5289438
```

0.1.3 Question 2c - Estimate ATE using doubly robust estimator

Estimate the ATE using the doubly robust estimator:

$$\theta_{\mathrm{ATE}_{\mathrm{DR}}} = \mathbb{E}\left[\frac{(Y - g(X,1))D}{p(X)} - \frac{(Y - g(X,0))(1-D)}{1-p(X)} + (g(X,1) - g(X,0))\right]$$

where the nuisance functions $p(X) = \mathbb{E}[D|X]$ and $g(X,D) = \mathbb{E}[Y|X,D]$ are estimated using random forest as well.

```r
rf_data <- subset(data2, select = c(-Y))

# Use RF to estimate p(X)
rf_prop <- randomForest(D ~ ., data = rf_data)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```r
rf_data$rf_prop_pred <- predict(rf_prop, rf_data)

# Append other vars
g1 <- rf_pred_treatment
g0 <- rf_pred_control
rf_data$Y <- data2$Y

# Trim the prop scores
rf_data <- rf_data %>%
    filter(rf_prop_pred > 0.01 & rf_prop_pred < 0.99)

# Calculate the doubly robust estimator for the ATE
doubly_robust_ate <- mean(((rf_data$Y - g1) * rf_data$D/rf_data$rf_prop_pred) -
    ((rf_data$Y - g0) * (1 - rf_data$D)/(1 - rf_data$rf_prop_pred)) +
    (g1 - g0))

# Output the ATE
doubly_robust_ate
```

```
## [1] 0.5267726
```

0.1.4 Question 2d - Estime ATE using cross-fitting

Estimate the ATE using the doubly robust estimator as before but compute the first step using cross-fitting for I = 2 sample split in the following way.

```
colnames(data2)  # has only V, D, Y
```

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12"
## [13] "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "D"   "Y"
```

```
# Divide sample into two folds
set.seed(123)  # Set seed for reproducibility
n <- nrow(data2)
id_div1 <- sample(1:n, n/2)
id_div2 <- setdiff(1:n, id_div1)
data_2d_DD_1 <- data2[id_div1, ]
data_2d_DD_2 <- data2[id_div2, ]

# Probit model for propensity score estimation
ps_estimator <- function(df) {
    probit_model <- glm(D ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 +
        V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 +
        V18 + V19 + V20, data = df, family = binomial(link = "probit"))
    df$ps_pred <- predict(probit_model, type = "response")
    df$ate_probit <- (df$D * df$Y/df$ps_pred) - ((1 - df$D) *
        df$Y/(1 - df$ps_pred))
    return(df)
}

# Naive estimation using random forests
theta_naive_fun <- function(df) {
    treatment <- subset(df, D == 1, select = -c(D, ps_pred, ate_probit))
    control <- subset(df, D == 0, select = -c(D, ps_pred, ate_probit))

    rf_treatment <- randomForest(Y ~ ., data = treatment)
    rf_control <- randomForest(Y ~ ., data = control)

    df$g0 <- predict(rf_control, df)
    df$g1 <- predict(rf_treatment, df)

    ate_naive <- mean(df$g1 - df$g0)
    return(df)
}

# Double Robustness (DR) estimation with trimming
theta_DR_trim <- function(df) {
    trimmed_df <- df %>%
        filter(ps_pred > 0.01 & ps_pred < 0.99) %>%
        mutate(left = (Y - g1) * D/ps_pred - (Y - g0) * (1 -
            D)/(1 - ps_pred), right = g1 - g0)
    return(trimmed_df)
}
```

```r
# Combined estimation function
theta_DD <- function(df) {
    df <- ps_estimator(df)
    ate_probit <- mean(df$ate_probit)

    df <- theta_naive_fun(df)
    ate_naive <- mean(df$g1 - df$g0)

    df <- theta_DR_trim(df)
    ate_DR <- mean(df$left + df$right)

    return(c(ate_probit, ate_naive, ate_DR))
}

# Apply estimation on the full data and folds
eta_j <- theta_DD(data2)
eta_1 <- theta_DD(data_2d_DD_1)
eta_2 <- theta_DD(data_2d_DD_2)

# Output results
list(eta_j, eta_1, eta_2)
```

```
## [[1]]
## [1] 0.4618182 0.5309806 0.4815573
##
## [[2]]
## [1] 0.4035959 0.3733229 0.3369678
##
## [[3]]
## [1] 0.4944975 0.7273893 0.6235399
```

```r
# Estimate ATE by DD (using DR estimations in each fold)
ate_DD <- (eta_1[3] + eta_2[3])/2
ate_DD
```

```
## [1] 0.4802538
```

0.1.5 Question 2e - Repeat process

Repeat process for 1000 Montecarlo simulations and compute the four estimators for each sample.

```r
# Generate the data
n_iter <- 1000

# Store bootstrap results
ate_list <- vector("list", length = n_iter)

# Iterating over bootstrap samples
for (i in 1:n_iter) {
    # Generate data for each iteration
    data_ex2_iter <- as.data.frame(MASS::mvrnorm(n = 500, mu = rep(0,
        20), Sigma = sigma))
    V <- V_fun(data_ex2_iter$V1, data_ex2_iter$V3)
    data_ex2_iter$D <- ifelse(V > 0, 1, 0)
    data_ex2_iter$Y <- Y_fun(data_ex2_iter$V1, data_ex2_iter$V3,
        data_ex2_iter$D)

    # Divide data into two groups
    id_div1_iter <- sample(1:length(data_ex2_iter$Y), length(data_ex2_iter$Y)/2,
        replace = FALSE)
    id_div2_iter <- setdiff(1:length(data_ex2_iter$Y), id_div1_iter)
    data_ex2_DD_1_iter <- data2[id_div1_iter, ]
    data_ex2_DD_2_iter <- data2[id_div2_iter, ]

    # Calculate coefficients
    eta_j_iter <- theta_DD(data_ex2_iter)
    eta_1_iter <- theta_DD(data_ex2_DD_1_iter)
    eta_2_iter <- theta_DD(data_ex2_DD_2_iter)
    ate_DD_iter <- (eta_1_iter[3] + eta_2_iter[3])/2

    # Append coefficients to the list
    all_coef <- append(eta_j_iter, ate_DD_iter)
    ate_list[[i]] <- all_coef
}
```

0.1.6 Question 2f - Compute empirical variance of each estimator

Compute the empirical variance of each of your estimators $\hat{\sigma}^2_{\theta_k}$ where k = probit, naive, DR, DD.

```r
ate_matrix <- do.call(rbind, ate_list)
variances <- apply(ate_matrix, 2, var)
variances
```

```
## [1] 0.0799602744 0.0119053229 0.0151571665 0.0005943325
```

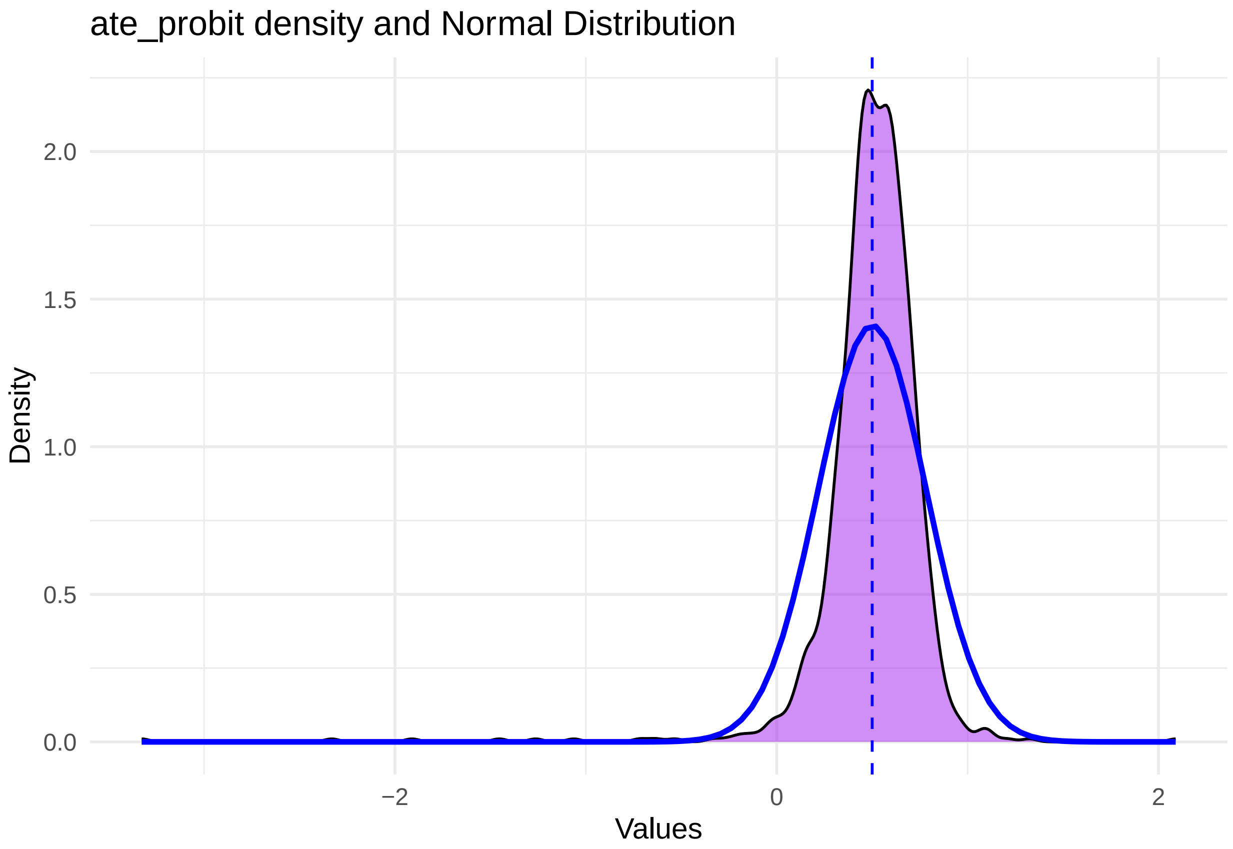
0.1.7 Question 2g - Plots

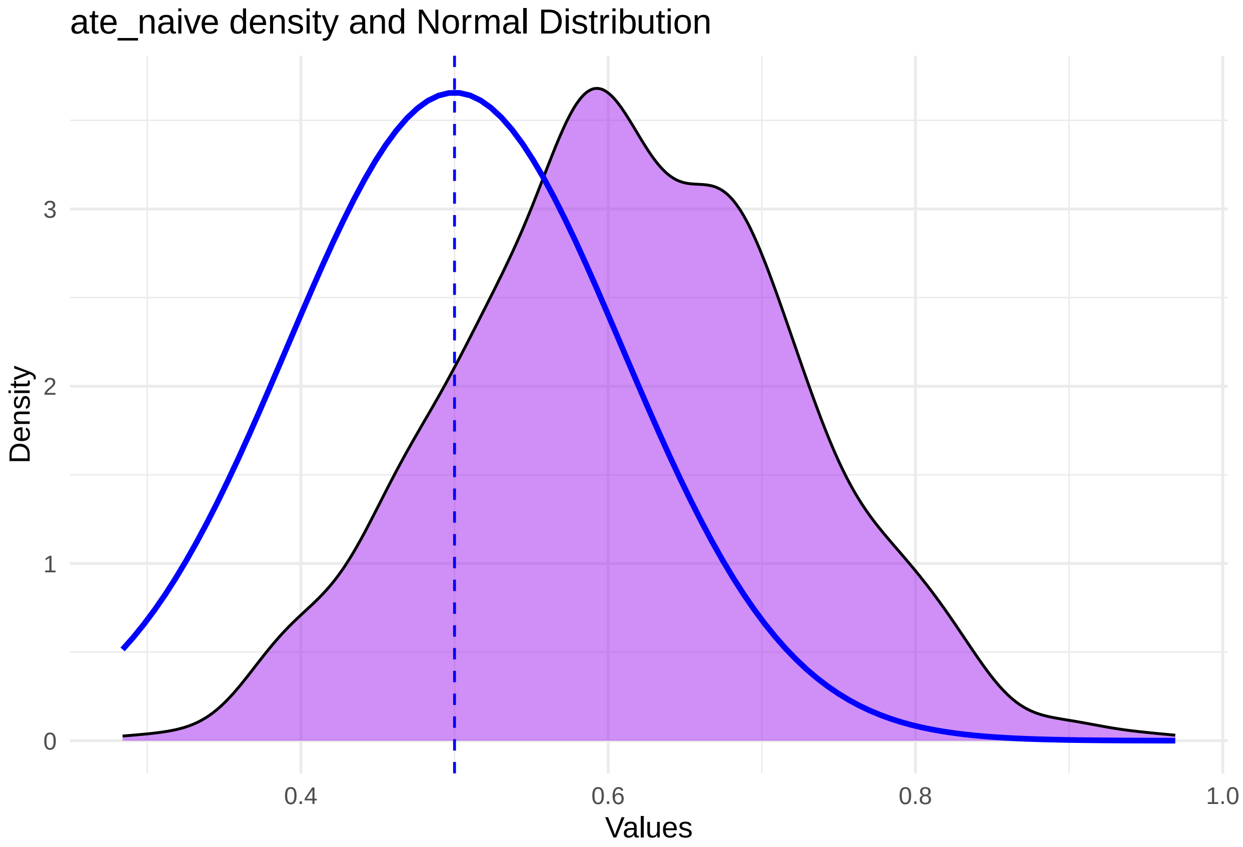
Create four plots, one for each estimator, containing the histogram of the $\hat{\theta}_{ATE_k}$ and a Normal(.5, $\hat{\sigma}^2_{\theta_k}$) pdf.

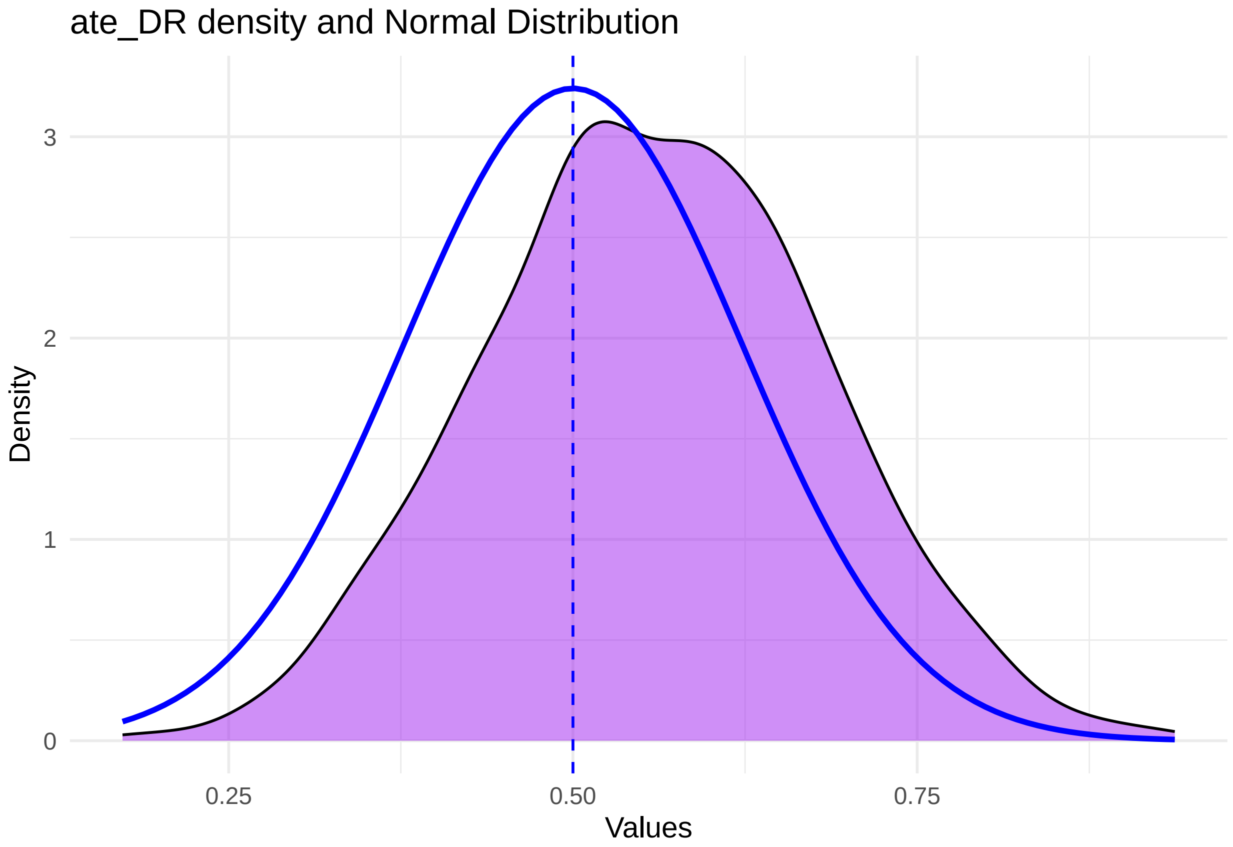
```r
df_ate <- as.data.frame(ate_matrix)
colnames(df_ate) <- c("ate_probit", "ate_naive", "ate_DR", "ate_DD")

# Define the names of the columns in df_ate
ate_names <- c("ate_probit", "ate_naive", "ate_DR", "ate_DD")

# Loop through each column name to create the plots
plots <- lapply(1:length(ate_names), function(i) {
    ggplot(df_ate, aes_string(x = ate_names[i])) + geom_density(fill = "purple",
        alpha = 0.5) + stat_function(fun = dnorm, args = list(mean = 0.5,
        sd = sqrt(variances[i])), color = "blue", linewidth = 1) +
        geom_vline(xintercept = 0.5, linetype = "dashed", color = "blue") +
        labs(title = paste(ate_names[i], "density and Normal Distribution"),
            x = "Values", y = "Density") + theme_minimal()
})
```

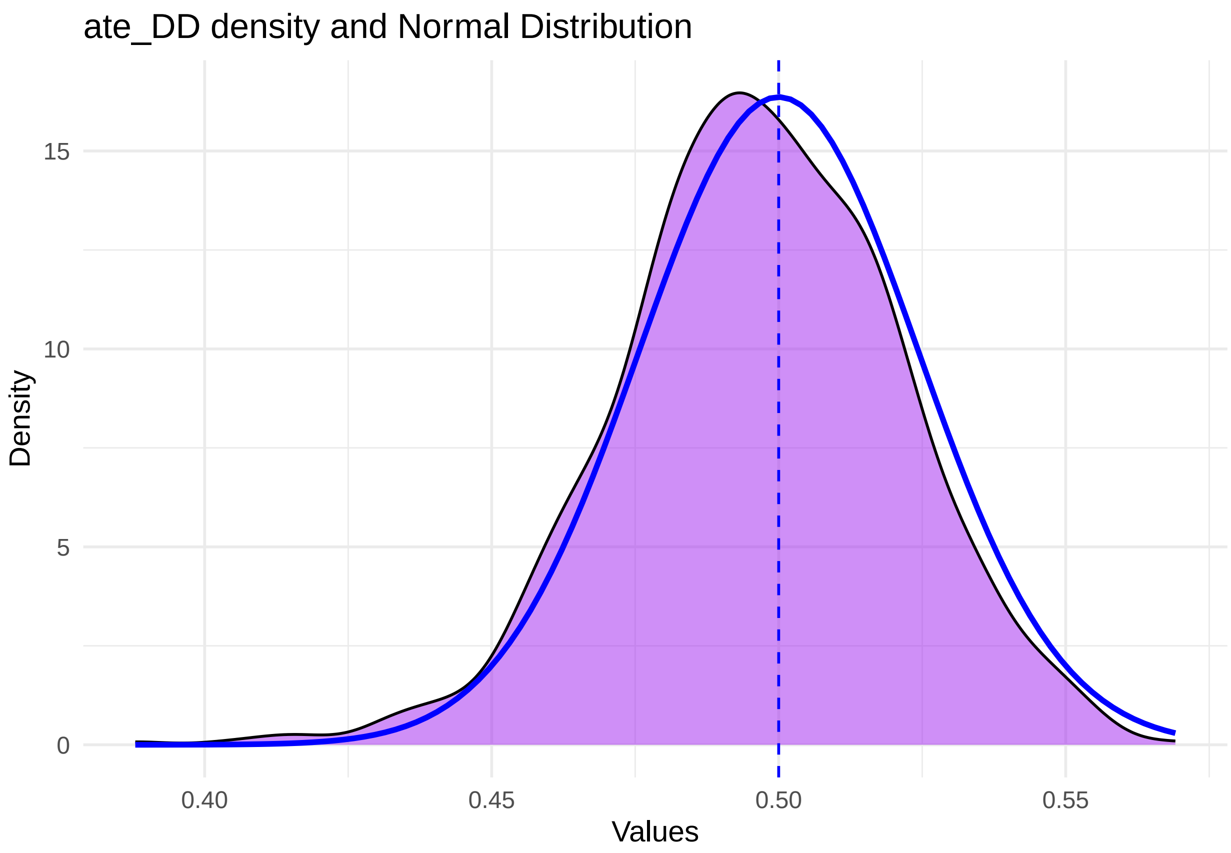
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
for (plot in plots) print(plot)
```




ate_naive density and Normal Distribution
Density
3
2
1
0
0.4
0.6
0.8
1.0
Values


ate_DR density and Normal Distribution
Density
3
2
1
0
0.25
0.50
0.75
Values



0.1.8 Question 2h - Table

Present a table comparing the bias and mean square error of each estimator and comment on which estimator performs better and why.

```
# Function to calculate MSE
calculate_mse <- function(n, mu, Sigma, method_ate) {
    observed <- as.data.frame(MASS::mvrnorm(n = n, mu = mu, Sigma = Sigma))
    return(mean((observed$V1 - method_ate)^2))
}

# Calculate bias
bias <- c(0.5, 0.5, 0.5, 0.5) - apply(ate_matrix, 2, mean)

# Initialize method names and corresponding ATEs
methods <- c("Probit", "Naive", "DR", "DD")
ates <- c(df_ate$ate_probit, df_ate$ate_naive, df_ate$ate_DR,
    df_ate$ate_DD)

# Calculate MSE for each method
mse_values <- sapply(1:4, function(i) calculate_mse(1000, 0.5,
    variances[i], ates[i]))

# Create a data frame
data <- data.frame(Method = methods, Bias = round(bias, 4), MSE = round(mse_values,
    4))

# Print the formatted table
knitr::kable(data, format = "markdown", caption = "Bias and MSE for Different Methods")
```

Table 1: Bias and MSE for Different Methods

Method	Bias	MSE
Probit	-0.0018	0.0812
Naive	-0.1118	0.0421
DR	-0.0582	0.0162
DD	0.0035	0.0297

I would prefer the DR estimator because it has the smallest MSE out of all the estimators, even though the probit has the smallest bias in absolute value. Since MSE is a loss function that minimizes $Var(T)+Bias(T)^2$ for an estimator T, minimizing MSE means that the DR estimator will be neither too biased nor have too large variance. This is compared to the probit estimator, which does has smaller bias, but it might come at the cost of higher variance.