POL SCI 231b: Section 7

GSI Guadalupe Tuñón

University of California, Berkeley

Spring 2016

Today

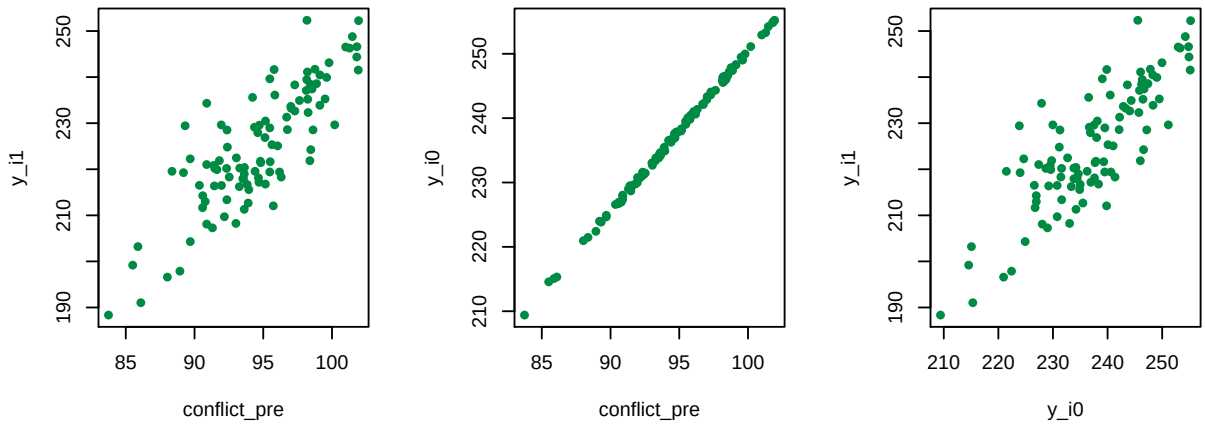
1. Variance of the Difference in Differences estimator
2. Randomization inference with covariates
3. Clustering
4. OLS vs fGLS - Huber-White SEs

Variance of the Difference in Differences estimator

A DGP (building losely from Miguel et. al).

```
# Pre-treatment covariates
conflict_pre <- rnorm(100, 95, 4)
rural <- rbinom(100, 1, 0.7)
ethnic_frac <- rnorm(100, 0.45, 0.3)

# Potential outcomes
y_i0 <- 2.5 * conflict_pre + runif(100)
y_i1 <- y_i0 - rnorm(100, 10, 5) + rnorm(100,
    4 * ethnic_frac, 3) - 8 * rural
```

```
par(mfrow = c(1, 3))
plot(conflict_pre, y_i1, pch = 16, col = "springgreen4")
plot(conflict_pre, y_i0, pch = 16, col = "springgreen4")
plot(y_i0, y_i1, pch = 16, col = "springgreen4")
```

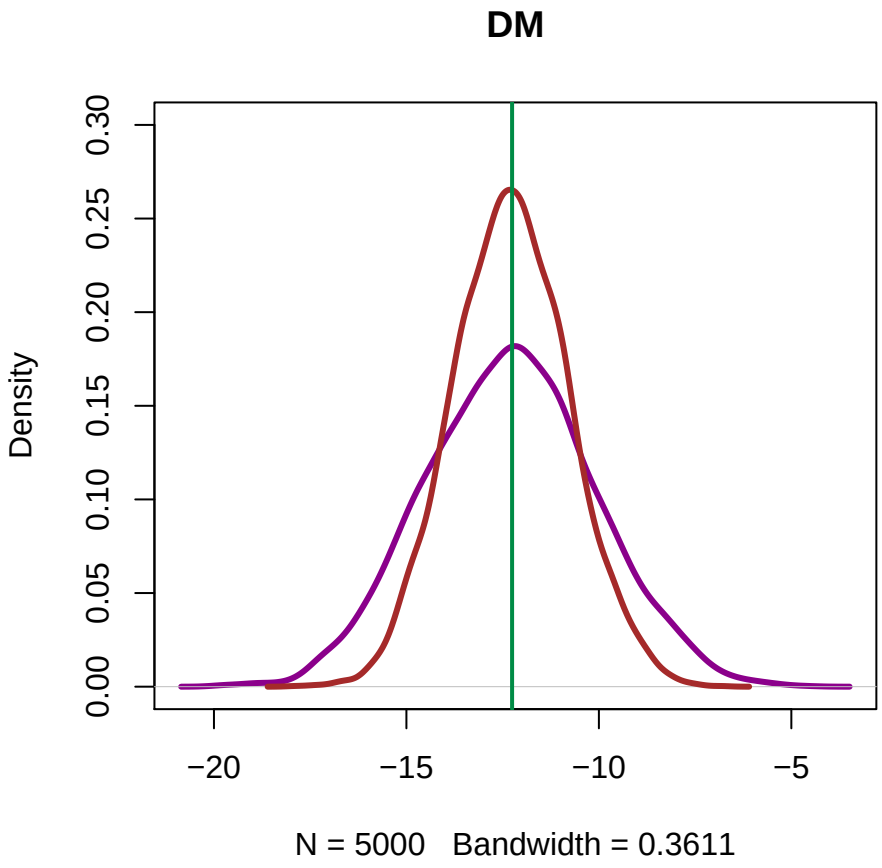



```
true_ace <- mean(y_i1) - mean(y_i0)
true_ace

## [1] -12.25455
```

A simulation

```
experiment <- function() {

    rainfall <- rbinom(100, 1, 0.5)
    observed_conflict <- ifelse(rainfall == 1,
        y_i1, y_i0)

    # simple difference of means
    DM <- mean(observed_conflict[rainfall == 1]) -
        mean(observed_conflict[rainfall == 0])

    # rescaling outcome (difference in
    # differences)
    rescaled <- observed_conflict - conflict_pre
    DiD <- mean(rescaled[rainfall == 1]) - mean(rescaled[rainfall ==
        0])

    # return
    return(c(DM, DiD))

}


reps <- replicate(5000, experiment())
rownames(reps) <- c("DM", "DiD")
```

```
plot(density(reps[1, ]), col = "darkmagenta",
    main = "DM", lwd = 3, ylim = c(0, 0.3))
lines(density(reps[2, ]), col = "brown", lwd = 3)
abline(v = true_ace, lwd = 2, col = "springgreen4")
```




```
apply(reps, MARGIN = 1, FUN = mean)

##         DM        DiD
## -12.30131 -12.28649

apply(reps, MARGIN = 1, FUN = sd)

##       DM      DiD
## 2.204130 1.489207
```

What can we say about the difference between these estimators? Is one of them "better"? If so, how so?

Our simulation shows the unbiasedness of the DiD estimator.

In terms of precision, we saw in lecture that the DiD estimator may be more precise, but that this is true only when the pre-treatment value of the outcome strongly predicts potential outcomes. Formally,

$$Cov(Y_i(0), X_i) + Cov(Y_i(1), X_i) > Var(X_i)$$

What are the values for these elements in our example?

```
var(conflict_pre)

## [1] 15.4925

cov(y_i1, conflict_pre)

## [1] 43.16722

cov(y_i0, conflict_pre)

## [1] 38.70579

cov(y_i1, conflict_pre) + cov(y_i0, conflict_pre) >
    var(conflict_pre)

## [1] TRUE
```

In your next problem set: How can we modify the simulation to show that the gains in precision depend on whether the pre-treatment value of the outcome strongly predicts potential outcomes?

Randomization inference with covariates (Rosenbaum 2002)

We first create a DGP with a set of covariates that are predictive of the potential outcomes.

```
library(mvtnorm)

# DGP
sigma <- matrix(1, 5, 5)
diag(sigma) <- 2
data <- as.data.frame(rmvnorm(n = 300, mean = rep(0,
    5), sigma = sigma))
names(data)[1] <- "r0"
data$r1 <- data$r0 + rnorm(300, 0.2, 1)  # ATE = .2
data$Z <- rbinom(300, 1, 0.5)
data$yobs <- ifelse(data$Z == 1, data$r1, data$r0)
```

We define a function that will produce residuals, $\epsilon(Y(0), X) = e$. The function could be pretty much anything as long as it was **definded before the treatment**. Here we will just a regression of Y (which is also Y(0) under the sharp null) on our four covariates.

```
e <- lm(data$yobs ~ data$V2 + data$V3 + data$V4 +
    data$V5)$residuals
```

We store the residuals from the regression, which will serve as our outcome variable.

We now get our test statistic of interest. Here we will just get the difference of means (remember regression is algebraically equivalent).

```
Tstat <- mean(e[data$Z == 1]) - mean(e[data$Z ==
    0])
Tstat

## [1] 0.2078875
```

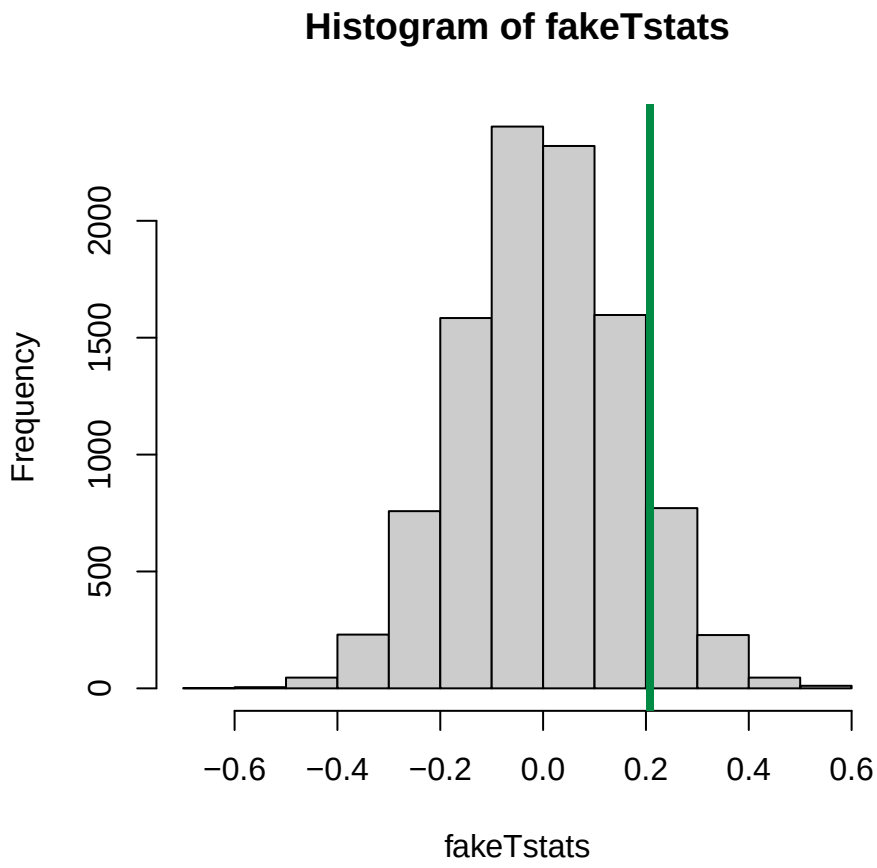
Now we need the randomization distribution. First we get the randomization matrix:

```
fakeZ <- matrix(NA, 300, 10000)
for (i in 1:10000) {
    fakeZ[, i] <- sample(data$Z, 300, replace = F)
}
```

And get a tstatistic for each random treatment vector.

```r
# function
DM <- function(e, z) {
    return(mean(e[z == 1]) - mean(e[z == 0]))
}

fakeTstats <- apply(fakeZ, 2, DM, e = e)
```

```r
hist(fakeTstats, col = "grey80")
abline(v = Tstat, col = "springgreen4", lwd = 4)
```



Finally, we calculate the p-value.

```r
pvalue <- sum(abs(fakeTstats) >= Tstat)/10000
pvalue

## [1] 0.192
```

Clustering

DGP

```
m <- 80  # cluster size
C <- 100  # nr of clusters
N <- m * C
cl <- rep(1:C, each = m)

y_0 <- c(rnorm(N/8, 0, 2), rnorm(N/4, 3, 2), rnorm(N/4,
    5, 1), rnorm(N/4, 7, 1), rnorm(N/8, 8, 2))
y_1 <- y_0 + rnorm(N, 3, 1)
```

What is the average causal effect?

```
ACE <- mean(y_1) - mean(y_0)
ACE

## [1] 3.000605
```

We will assign 4000 units to treatment and 4000 to control

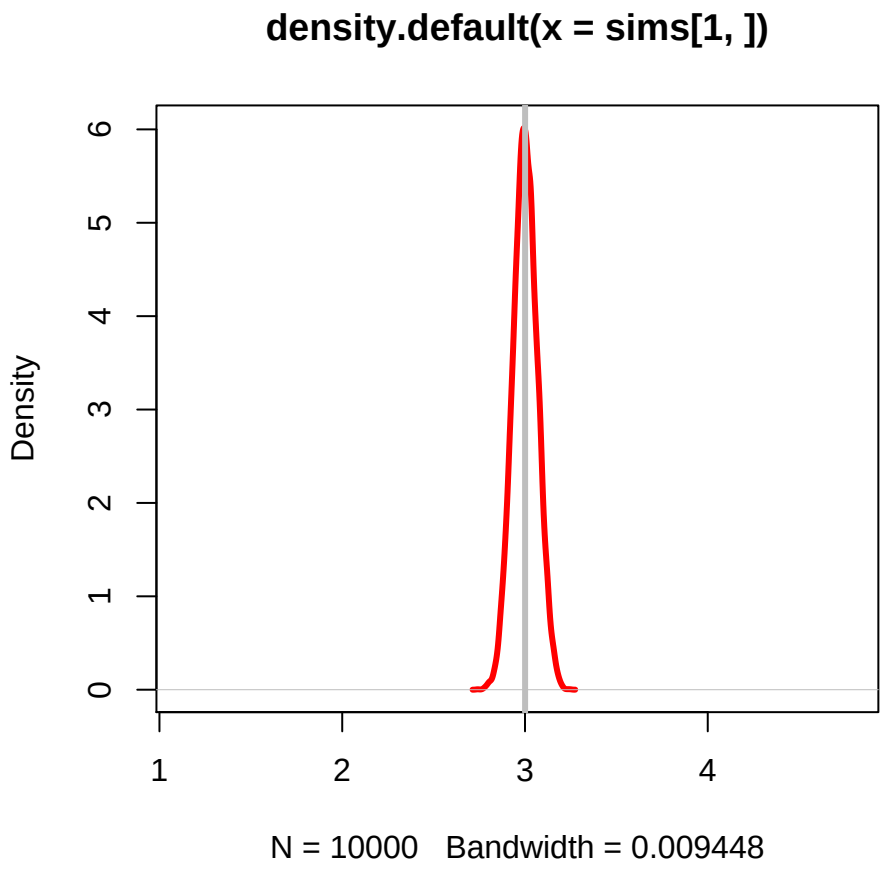
```
simulation <- function() {

    # simple random sampling
    simple_random <- sample(c(rep(1, N/2), rep(0,
        N/2)), N, replace = F)

    # cluster random sampling
    clusters_treat <- sample(1:C, C/2, replace = FALSE)
    g <- function(x) {
        sum(x == clusters_treat)
    }
    cluster_random <- unlist(lapply(cl, FUN = g))

    # diff in means simple random assignment
    dm_sr <- mean(y_1[simple_random == 1]) - mean(y_0[simple_random ==
        0])

    # diff in means cluster assignment, individual
    dm_cl <- mean(y_1[cluster_random == 1]) -
        mean(y_0[cluster_random == 0])
```

```
    # diff in means cluster assignment, cluster
    # means
    cl_treat <- unlist(lapply(clusters_treat,
        FUN = function(x) mean(y_1[cl == x])))
    cl_control <- unlist(lapply(as.vector(1:C)[-(clusters_treat)],
        FUN = function(x) mean(y_0[cl == x])))
    dm_cl_cl <- mean(cl_treat) - mean(cl_control)

    return(c(dm_sr, dm_cl, dm_cl_cl))

}
```

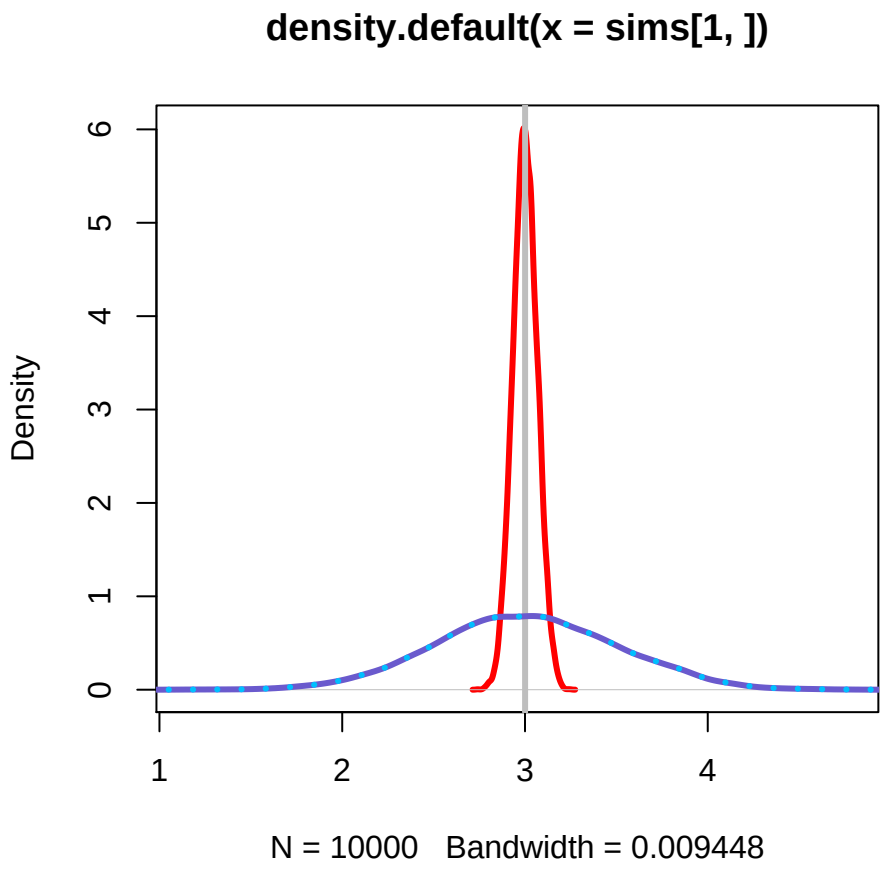
```
simulation()

## [1] 2.907437 2.732269 2.732269

sims <- replicate(10000, simulation())
```

How do we expect the results to look like?

```
plot(density(sims[1, ]), col = "red", lwd = 3,
    xlim = c(min(sims), max(sims)))
abline(v = ACE, col = "grey", lwd = 3)
```




```
plot(density(sims[1, ]), col = "red", lwd = 3,
    xlim = c(min(sims), max(sims)))
abline(v = ACE, col = "grey", lwd = 3)
lines(density(sims[2, ]), col = "slateblue", lwd = 3)
lines(density(sims[3, ]), col = "deepskyblue",
    lwd = 3, lty = 3)
```




```
apply(sims, MARGIN = 1, FUN = mean)

## [1] 3.001028 3.008015 3.008015

apply(sims, MARGIN = 1, FUN = sd)

## [1] 0.0662361 0.4962054 0.4962054
```

OLS and Heteroscedasticity

The problem:

Setup

```
y_0 <- rnorm(100, 0, 1)
y_1 <- y_0 + rnorm(100, 2, 10)

sd(y_0)

## [1] 0.9669063

sd(y_1)

## [1] 9.895268

treat <- rbinom(100, 1, 0.5)
Y <- ifelse(treat == 1, y_1, y_0)
```

```
source("~/Dropbox/Resources/r_functions/t_test.R")
ttest(Y, treat)

##       Mean 1       Mean 0   Difference
##   0.86621324  -0.02155845   0.88777170
##      SE_Diff       t-stat            N
##   1.32144219   0.67182031 100.00000000
##           df      p-value
##  44.68175096   0.50515711

summary(lm(Y ~ treat))

##
## Call:
## lm(formula = Y ~ treat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.5782  -1.0053  -0.0571   1.0120  23.4088
##
## Coefficients:
##             Estimate Std. Error t value
## (Intercept) -0.02156    0.80245  -0.027
```

```
## treat        0.88777    1.19622   0.742
##             Pr(>|t|)
## (Intercept)    0.979
## treat          0.460
##
## Residual standard error: 5.951 on 98 degrees of freedom
## Multiple R-squared:  0.005589,Adjusted R-squared:  -0.004558
## F-statistic: 0.5508 on 1 and 98 DF,  p-value: 0.4598
```

As you have shown in your problem sets, it will often be the case in experiments that

$$E(\epsilon|T) \neq \sigma^2$$

Indeed,

$$E(\epsilon_i|T_i = 1) = Var(Y_i(1))$$

$$E(\epsilon_i|T_i = 0) = Var(Y_i(0))$$

which are only equal when

$$Var(Y_i(1)) = Var(Y_i(0))$$

Back to section 4: **How do we calculate $SE(\hat{\beta})$ when we assume homoskedasticity?**

```
# The first thing we will need here is to
# build the matrix for X
X <- cbind(1, treat)

# getting the betas
betas <- solve(t(X) %*% X) %*% t(X) %*% Y

# and the residuals
e <- Y - X %*% betas

# and now we want the estimated sigma squared
hat_sigma2 <- sum(t(e) %*% e)/(nrow(X) - length(betas))

# and we can find estimated standard errors
# from $\hat{\sigma^2}(X'X)^{-1}$.
var_hat_beta <- hat_sigma2 * solve((t(X) %*% X))  # Why do we use * instead of %*% her

# and get the SEs for our betas
se_hat_beta <- sqrt(diag(var_hat_beta))
se_hat_beta
```

```
##               treat
## 0.8024517 1.1962244
```

The White-Huber correction

$$\hat{cov}(\hat{\beta}) = (X'X)^{-1}(X'\hat{G}X)(X'X)^{-1}$$

The shortcut

```
library(sandwich)
fit <- lm(Y ~ treat)
sqrt(vcovHC(fit, type = "HC0")[2, 2])

## [1] 1.306698
```

Let's go through what happens under the hood:
First we need the "bread": $(X'X)$

```
bread <- solve(t(X) %*% X)
```

And "butter":

```
Ghat <- diag(fit$residuals)^2

butter <- t(X) %*% Ghat %*% X


EHW <- bread %*% butter %*% bread
```

What elements from this matrix do we need?

```
sqrt(EHW[2, 2])

## [1] 1.306698
```

Let's compare this to the results from the t-test:

```
ttest(Y, treat)

##       Mean 1       Mean 0   Difference
##   0.86621324  -0.02155845   0.88777170
##      SE_Diff       t-stat            N
##   1.32144219   0.67182031 100.00000000
##           df      p-value
##  44.68175096   0.50515711
```