Homework 6 Solutions

Hansen 17.2

$\mathrm{E}[e_{it}|X_{it}] = 0$ is not a strong enough condition for $\hat{\beta}$ from a fixed effects regression to be unbiased. This condition says that e_{it} is (mean) independent of X_{it}, but it does not rule out that e_{it} could be related to, say, X_{it+1}. This is not an entirely strange case either, particularly if a good "shock" in the current period leads to the covariate changing in the next time period.

More specifically, recall that

$$\begin{aligned}
\mathrm{E}[\hat{\beta} - \beta|\mathbf{X}] &= \left(\sum_{i=1}^{n} \dot{\mathbf{X}}_i' \dot{\mathbf{X}}_i\right)^{-1} \sum_{i=1}^{n} \dot{\mathbf{X}}_i' \mathrm{E}[\mathbf{e}_i|\mathbf{X}] \\
&= \left(\sum_{i=1}^{n} \dot{\mathbf{X}}_i' \dot{\mathbf{X}}_i\right)^{-1} \sum_{i=1}^{n} \dot{\mathbf{X}}_i' \mathrm{E}[\mathbf{e}_i|\mathbf{X}_i]
\end{aligned}$$

where $\mathbf{X}$ is the $nT \times k$ "data matrix" and the other notation is from class, and the second equality uses that the observations are independent of each other. Notice that,

$$\mathrm{E}[\mathbf{e}_i|\mathbf{X}_i] = \begin{bmatrix} \mathrm{E}[e_{i1}|X_{i1}, X_{i2}, \ldots, X_{iT}] \\ \mathrm{E}[e_{i2}|X_{i1}, X_{i2}, \ldots, X_{iT}] \\ \vdots \\ \mathrm{E}[e_{iT}|X_{i1}, X_{i2}, \ldots, X_{iT}] \end{bmatrix}$$

The condition in the problem is not strong enough that this term is equal to 0. And, if it is some function of X, then $\hat{\beta}$ would not, in general, be unbiased for β.

Additional Question 2

Yes, they are correct. Notice that, because there are only two periods, this regression is equivalent to

$$\Delta Y_{i2} = \Delta\theta_2 + \alpha D_{i2} + \Delta e_{i2}$$

Moreover,

$$\mathrm{E}[\Delta Y_2|D_2 = 1] = \Delta\theta_2 + \alpha \tag{1}$$

$$\mathrm{E}[\Delta Y_2|D_2 = 0] = \Delta\theta_2 \tag{2}$$

which implies that

$$\begin{aligned}\alpha &= \mathrm{E}[\Delta Y_2|D_2=1] - \mathrm{E}[\Delta Y_2|D_2=0] \\ &= \mathrm{E}[\Delta Y_2|D_2=1] - \mathrm{E}[\Delta Y_2(0)|D_2=0] \\ &= \mathrm{E}[\Delta Y_2|D_2=1] - \mathrm{E}[\Delta Y_2(0)|D_2=1] \\ &= ATT\end{aligned}$$

where the first line subtracts Equation 2 from Equation 1, the second equality writes the second term in terms of potential outcomes, and the third equality holds by the parallel trends assumption.

Additional Question 2

```
set.seed(1234)
library(randomForest)
data <- as.data.frame(haven::read_dta("jtrain_observational.dta"))

n <- nrow(data)
data$id <- 1:n
fold1 <- subset(data, (id%%2) == 0)
fold2 <- subset(data, (id%%2) == 1)

ml_att <- function(f1, f2) {

  # use f1 to estimate the first step models
  Dmod <- randomForest(as.factor(train) ~ age + educ + black + hisp +
                         married + re75 + unem75, data=f1)
  Ymod <- randomForest(re78 ~ age + educ + black + hisp +
                         married + re75 + unem75, data=f1)

  # get predictions with f2
  pscore <- predict(Dmod, newdata=f2, type="prob")[,2] # this gets p(d=1|x)
  out_reg <- predict(Ymod, newdata=f2)

  # compute att(k) with f2
  D <- f2$train
  p <- mean(D)
  Y <- f2$re78
  att1 <- mean(D/p*(Y-out_reg))
  att2 <- mean((1-D)/p * pscore/(1-pscore) * (Y-out_reg) )
```

```
  att <- att1-att2
  att
}

# cross splitting
ml1 <- ml_att(fold1,fold2)
# reverse roles
ml2 <- ml_att(fold2,fold1)
# average
mean(c(ml1,ml2))
```

```
## [1] 1.580708
```

This is larger than what we estimated on the last homework (about 0.85) though I noticed that my estimates do move somewhat if I run the code multiple times.

Additional Question 3

Part (a)

```
library(Matrix)
load("job_displacement_clean2.RData")
# drop already treated
data <- subset(data, first.displaced != 2001)
data <- droplevels(data)
Y <- data$learn
data$D <- 1*( (data$year >= data$first.displaced) & data$first.displaced != 0)
X <- model.matrix(~ as.factor(year) + D, data=data)
n <- length(unique(data$id))
tp <- length(unique(data$year))
iT <- matrix(rep(1,tp))
Dmat <- bdiag(replicate(n,iT,simplify=FALSE))
M <- Matrix::Diagonal(n*tp) - Dmat%*%solve(t(Dmat)%*%Dmat)%*%t(Dmat)
bet <- solve(t(X)%*%M%*%X) %*% t(X)%*%M%*%Y
bet
```

```
## 8 x 1 Matrix of class "dgeMatrix"
##              [,1]
## [1,]  5.29358972
## [2,]  0.09436799
```

```
## [3,]  0.18195412
## [4,]  0.26904810
## [5,]  0.28752717
## [6,]  0.35242722
## [7,]  0.38427488
## [8,] -0.23557976
```

Next, let's calculate the standard errors where we use that

$$\begin{aligned}\sqrt{n}(\hat{\beta} - \beta) &= \mathrm{E}[\mathbf{X}_i'\mathbf{M}_i\mathbf{X}_i]^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\mathbf{X}_i'\mathbf{M}_i\mathbf{e}_i + o_p(1) \\ &\xrightarrow{d} N(0, \mathbf{V})\end{aligned}$$

where

$$\begin{aligned}\mathbf{V} &= \mathrm{E}[\mathbf{X}_i'\mathbf{M}_i\mathbf{X}_i]^{-1}\boldsymbol{\Omega}\mathrm{E}[\mathbf{X}_i'\mathbf{M}_i\mathbf{X}_i]^{-1} \\ &= \mathrm{E}[\dot{\mathbf{X}}_i'\dot{\mathbf{X}}_i]^{-1}\boldsymbol{\Omega}\mathrm{E}[\dot{\mathbf{X}}_i'\dot{\mathbf{X}}_i]^{-1}\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{\Omega} &= \mathrm{E}[\mathbf{X}_i'\mathbf{M}_i\mathbf{e}_i\mathbf{e}_i'\mathbf{M}_i\mathbf{X}_i] \\ &= \mathrm{E}[\dot{\mathbf{X}}_i'\mathbf{e}_i\mathbf{e}_i'\dot{\mathbf{X}}_i]\end{aligned}$$

It's worth thinking about how to actually estimate these because $\dot{\mathbf{X}}_i$ is a matrix rather than our usual case of it being a vector. First, notice that

$$\dot{\mathbf{X}}_i'\dot{\mathbf{X}}_i = \sum_{t=1}^{T}\dot{X}_{it}\dot{X}_{it}'$$

which is a $k \times k$ matrix. Thus, the natural estimate of $\mathrm{E}[\dot{\mathbf{X}}_i'\dot{\mathbf{X}}_i]$ is

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{T}\dot{X}_{it}\dot{X}_{it}' = \dot{\mathbf{X}}'\dot{\mathbf{X}}/n$$

which corresponds to exactly the same way that we estimated this type of term throughout the semester. Next,

$$\dot{\mathbf{X}}_i'\mathbf{e}_i = \sum_{t=1}^{T}X_{it}e_{it}$$

which is a $k \times 1$ vector and so that

$$\dot{\mathbf{X}}_i' \mathbf{e}_i \mathbf{e}_i' \dot{\mathbf{X}}_i = \left(\sum_{t=1}^{T} \dot{X}_{it} e_{it} \right) \left(\sum_{t=1}^{T} \dot{X}_{it} e_{it} \right)'$$

and implies that we would estimate $\boldsymbol{\Omega}$ by

$$\hat{\boldsymbol{\Omega}} = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{t=1}^{T} \dot{X}_{it} \hat{e}_{it} \right) \left(\sum_{t=1}^{T} \dot{X}_{it} \hat{e}_{it} \right)'$$

As far as I know, you can't play the same matrix algebra "trick" that we usually use here (in particular, recall that in the cross sectional case we could estimate $\hat{\boldsymbol{\Omega}} = \frac{1}{n} \sum_{i=1}^{n} X_i X_i' \hat{e}_i^2$, but that, for programming, it was often convenient to re-express this $\hat{\boldsymbol{\Omega}} = \tilde{\mathbf{X}}' \tilde{\mathbf{X}} / n$ where a typical element of $\tilde{\mathbf{X}}$ is given by $X_i \hat{e}_i$.) Anyway, the line below that uses the `rowsum` function is essentially just manually calculating $\sum_{t=1}^{T} X_{it} \hat{e}_{it}$ and then using matrix algebra below it.

```
ehat <- as.numeric(Y - X%*%bet)
n <- length(unique(data$id))
dotX <- M%*%X
Q <- t(X) %*% dotX / n
dotXe <- rowsum(as.matrix(dotX*ehat), group=data$id)
Omeg <- t(dotXe)%*%dotXe/n
V <- solve(Q)%*%Omeg%*%solve(Q)
se <- sqrt(diag(V))/sqrt(n)
round(cbind.data.frame(bet=as.numeric(bet), se=se), 4)
```

```
##       bet     se
## 1  5.2936 0.1003
## 2  0.0944 0.0085
## 3  0.1820 0.0098
## 4  0.2690 0.0108
## 5  0.2875 0.0115
## 6  0.3524 0.0116
## 7  0.3843 0.0124
## 8 -0.2356 0.0257
```

Thus, we estimate that job displacement reduces earnings by about 23%. As a check, let's compare this to what we get from `fixest`.

```
library(fixest)
fe_reg <- feols(learn ~ as.factor(year) + D | id, data=data)
summary(fe_reg)
```

```
## OLS estimation, Dep. Var.: learn
## Observations: 18,928
## Fixed-effects: id: 2,704
## Standard-errors: Clustered (id)
##                      Estimate Std. Error  t value  Pr(>|t|)
## as.factor(year)2003  0.094368   0.008533 11.05873 < 2.2e-16 ***
## as.factor(year)2005  0.181954   0.009756 18.65073 < 2.2e-16 ***
## as.factor(year)2007  0.269048   0.010758 25.00982 < 2.2e-16 ***
## as.factor(year)2009  0.287527   0.011507 24.98624 < 2.2e-16 ***
## as.factor(year)2011  0.352427   0.011598 30.38799 < 2.2e-16 ***
## as.factor(year)2013  0.384275   0.012396 30.99875 < 2.2e-16 ***
## D                   -0.235580   0.025677 -9.17486 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.352029     Adj. R2: 0.75681
##                  Within R2: 0.107896
```

These appear to be the same (or the same up to possibly a degree of freedom adjustment).

Part (b)

```r
# list of time periods
# for simplicity I'm going to convert this to 1,2,3,4,5,6,7...
tlist <- (sort(unique(data$year)) - 1999)/2
# list of groups (excluding never-treated)
glist <- (sort(unique(data$first.displaced))[-1] - 1999)/2

# create new variables in updated time scale
data$G <- ifelse(data$first.displaced==0, 0, (data$first.displaced - 1999)/2)
data$tp <- (data$year-1999)/2



# write a function to compute att(g,t)
# I compute these as averages using weights, but it
# is fine to use subsets of data here too.
# @param w weights, used for bootstrap
# @param base_period, allows base period to optionally
#  be fixed at one
compute.attgt <- function(data, w=rep(1,nrow(data)),
                          use_base_period_1=FALSE) {
```

```r
  # data frame to store results
  results <- list()
  counter <- 1

  for (this_t in tlist[-1]) {
    for (this_g in glist) {
      base_period <- min(this_t-1, this_g-1)
      if (use_base_period_1) base_period <- 1
      G <- 1*(data$G==this_g)
      U <- 1*(data$G==0)
      pre <- 1*(data$tp == base_period)
      post <- 1*(data$tp == this_t)
      pg <- weighted.mean(data$G == this_g, w=w)
      pu <- weighted.mean(data$G == 0, w=w)
      ppre <- mean(pre)
      ppost <- mean(post)

      this_attgt <- weighted.mean(data$learn*G*post/pg/ppost, w=w) -
                        weighted.mean(data$learn*G*pre/pg/ppre, w=w) -
        (weighted.mean(data$learn*U*post/pu/ppost, w=w) -
           weighted.mean(data$learn*U*pre/pu/ppre, w=w))
      results[[counter]] <- c(attgt=this_attgt, g=this_g, t=this_t)

      counter <- counter+1
    }
  }

  # convert to data frame
  results <- as.data.frame(do.call("rbind", results))

  results
}

results <- compute.attgt(data)
# print results
round(results[order(results$g, results$t),],4)
```

```
##      attgt g t
## 1  -0.2091 2 2
```

```
## 7  -0.1562 2 3
## 13 -0.1775 2 4
## 19 -0.2375 2 5
## 25 -0.2347 2 6
## 31 -0.2781 2 7
## 2   0.0117 3 2
## 8  -0.1138 3 3
## 14 -0.1124 3 4
## 20 -0.1677 3 5
## 26 -0.1698 3 6
## 32 -0.0313 3 7
## 3   0.0726 4 2
## 9  -0.0641 4 3
## 15 -0.1989 4 4
## 21 -0.3070 4 5
## 27 -0.2000 4 6
## 33 -0.2506 4 7
## 4  -0.0128 5 2
## 10  0.0036 5 3
## 16 -0.0603 5 4
## 22 -0.3184 5 5
## 28 -0.3115 5 6
## 34 -0.2210 5 7
## 5   0.0195 6 2
## 11 -0.1013 6 3
## 17 -0.0129 6 4
## 23  0.0565 6 5
## 29 -0.2505 6 6
## 35 -0.1633 6 7
## 6   0.1183 7 2
## 12 -0.0379 7 3
## 18 -0.0295 7 4
## 24  0.0205 7 5
## 30 -0.1143 7 6
## 36 -0.2056 7 7
```

Part (c)

```r
# function to compute att0
# ret_weights argument optionally returns the underlying
# weights rather than att0
compute.att0 <- function(attgt_results, w=rep(1,nrow(data)),
                         ret_weights=FALSE) {
  # overall attgt weights
  ever_treated <- which(data$G != 0)
  w <- w[ever_treated]
  pg <- sapply(glist, function(g) weighted.mean(data[ever_treated,]$G==g, w=w))
  maxT <- max(tlist)
  w0 <- function(g,t) {
    1*(t >= g)*pg[glist==g] / (maxT - g + 1)
  }
  # add weights to results
  w0gt <- sapply(1:nrow(attgt_results),
                 function(i) w0(attgt_results$g[i], attgt_results$t[i]))
  attgt_results$w0 <- w0gt
  # optionally return computed weights
  if(ret_weights) return(attgt_results)
  att0 <- sum(attgt_results$attgt*attgt_results$w0)
  att0
}

att0 <- compute.att0(results)

# bootstrap standard errors
B <- 100
id_list <- unique(data$id)
boot_att0 <- list()
for (b in 1:B) {
  # draw weights from multinomial distribution (this is exactly the same
  # as empirical bootstrap)
  boot_weights <- as.numeric(rmultinom(n=1, size=n, prob=rep(1/n,n)))
  this_boot_weights_id <- cbind.data.frame(id=id_list, boot_weights=boot_weights)
  boot_data <- merge(data, this_boot_weights_id, by="id")
  boot_attgt <- compute.attgt(data, w=boot_data$boot_weights)
  boot_att0[[b]] <- compute.att0(boot_attgt, w=boot_data$boot_weights)
```

```r
}

boot_att0 <- do.call("rbind", boot_att0)
se <- sd(boot_att0)

round(cbind.data.frame(att0=att0, se=se), 4)
```

```
##      att0     se
## 1 -0.2111 0.0236
```

Thus, we are estimating a large, negative and statistically significant effect of job displacement.

Part (d)

```r
# twfe weights
Edt <- function(t) {
  mean( (t >= data$G) & (data$G !=0) )
}
mEdt <- mean(sapply(tlist, Edt))
pg2 <- sapply(glist, function(g) mean(data$G==g))
maxT <- max(tlist)
wTWFE_num <- function(g,t,post0=FALSE) {
  if ((t < g) & post0) return(0)
  hgt <- 1*(t>=g) - (maxT-g+1)/maxT - Edt(t) + mEdt
  hgt*pg2[glist==g]
}
# add weights to results
wTWFEgt_num <- sapply(1:nrow(results),
                      function(i) wTWFE_num(results$g[i],
                                            results$t[i],
                                            post0=TRUE))
wTWFEgt_den <- sapply(1:nrow(results),
                      function(i) wTWFE_num(results$g[i],
                                            results$t[i],
                                            post0=TRUE))
wTWFEgt <- wTWFEgt_num/sum(wTWFEgt_den)
results$wTWFE <- wTWFEgt

# get results from att0
```

```r
results$watt0 <- compute.att0(results, ret_weights=TRUE)$w0

# print results
round(results[order(results$g, results$t),], 4)
```

```
##       attgt g t  wTWFE  watt0
## 1  -0.2091 2 2 0.0394 0.0378
## 7  -0.1562 2 3 0.0330 0.0378
## 13 -0.1775 2 4 0.0284 0.0378
## 19 -0.2375 2 5 0.0218 0.0378
## 25 -0.2347 2 6 0.0171 0.0378
## 31 -0.2781 2 7 0.0105 0.0378
## 2   0.0117 3 2 0.0000 0.0000
## 8  -0.1138 3 3 0.0465 0.0344
## 14 -0.1124 3 4 0.0430 0.0344
## 20 -0.1677 3 5 0.0380 0.0344
## 26 -0.1698 3 6 0.0344 0.0344
## 32 -0.0313 3 7 0.0294 0.0344
## 3   0.0726 4 2 0.0000 0.0000
## 9  -0.0641 4 3 0.0000 0.0000
## 15 -0.1989 4 4 0.0454 0.0303
## 21 -0.3070 4 5 0.0419 0.0303
## 27 -0.2000 4 6 0.0394 0.0303
## 33 -0.2506 4 7 0.0358 0.0303
## 4  -0.0128 5 2 0.0000 0.0000
## 10  0.0036 5 3 0.0000 0.0000
## 16 -0.0603 5 4 0.0000 0.0000
## 22 -0.3184 5 5 0.0836 0.0594
## 28 -0.3115 5 6 0.0799 0.0594
## 34 -0.2210 5 7 0.0748 0.0594
## 5   0.0195 6 2 0.0000 0.0000
## 11 -0.1013 6 3 0.0000 0.0000
## 17 -0.0129 6 4 0.0000 0.0000
## 23  0.0565 6 5 0.0000 0.0000
## 29 -0.2505 6 6 0.0718 0.0626
## 35 -0.1633 6 7 0.0682 0.0626
## 6   0.1183 7 2 0.0000 0.0000
## 12 -0.0379 7 3 0.0000 0.0000
## 18 -0.0295 7 4 0.0000 0.0000
```

```
## 24  0.0205 7 5 0.0000 0.0000
## 30 -0.1143 7 6 0.0000 0.0000
## 36 -0.2056 7 7 0.1178 0.1761
```

Notice that none of the TWFE weights are negative here though some of them do seem fairly different from the weights on ATT^O.

As a final side-comment, you might notice that

```
attTWFE <- sum(results$attgt*results$wTWFE)
attTWFE
```

```
## [1] -0.2139115
```

is not exactly equal to α that we calculated earlier. There are two reasons for this. First, our expression for α in terms of underlying $ATT(g,t)$'s relied on parallel trends actually holding; so if it does not, then we will not get exactly the same thing. Second, there is estimation error in $ATT(g,t)$; that is, we have $\widehat{ATT}(g,t)$ rather than $ATT(g,t)$, and the way to estimate this is not unique. Let me very quickly give show how you can recover α. In the notes, the line right before relating α to underlying $ATT(g,t)$'s was

$$\alpha = \sum_{t=2}^{T} \sum_{g \in \bar{\mathcal{G}}} h(g,t) \Big(\mathrm{E}[Y_{it} - Y_{i1}) | G = g] - \mathrm{E}[Y_{it} - Y_{i1}) | U = 0] \Big) p_g \Big/ \sum_{t=1}^{T} \mathrm{E}[\ddot{D}_{it}^2]$$

We can use this to compute a "decomposition" of α that will be equal to what we actually estimated.

```
# this computes "ATT(g,t)'s" using base period = 1
# everywhere which is analogous to above equation
results2 <- compute.attgt(data, use_base_period_1=TRUE)
# compute weights but allow for non-zero weights
# in pre-treatment periods
wTWFEgt_num2 <- sapply(1:nrow(results),
                       function(i) wTWFE_num(results$g[i],
                                             results$t[i],
                                             post0=FALSE))
wTWFEgt_den2 <- sapply(1:nrow(results),
                       function(i) wTWFE_num(results$g[i],
                                             results$t[i],
                                             post0=TRUE))
wTWFEgt2 <- wTWFEgt_num2/sum(wTWFEgt_den2)
# check if this delivers alpha
sum(results2$attgt*wTWFEgt2)
```

```
## [1] -0.2355798
```

which are now the same.

Part (e)

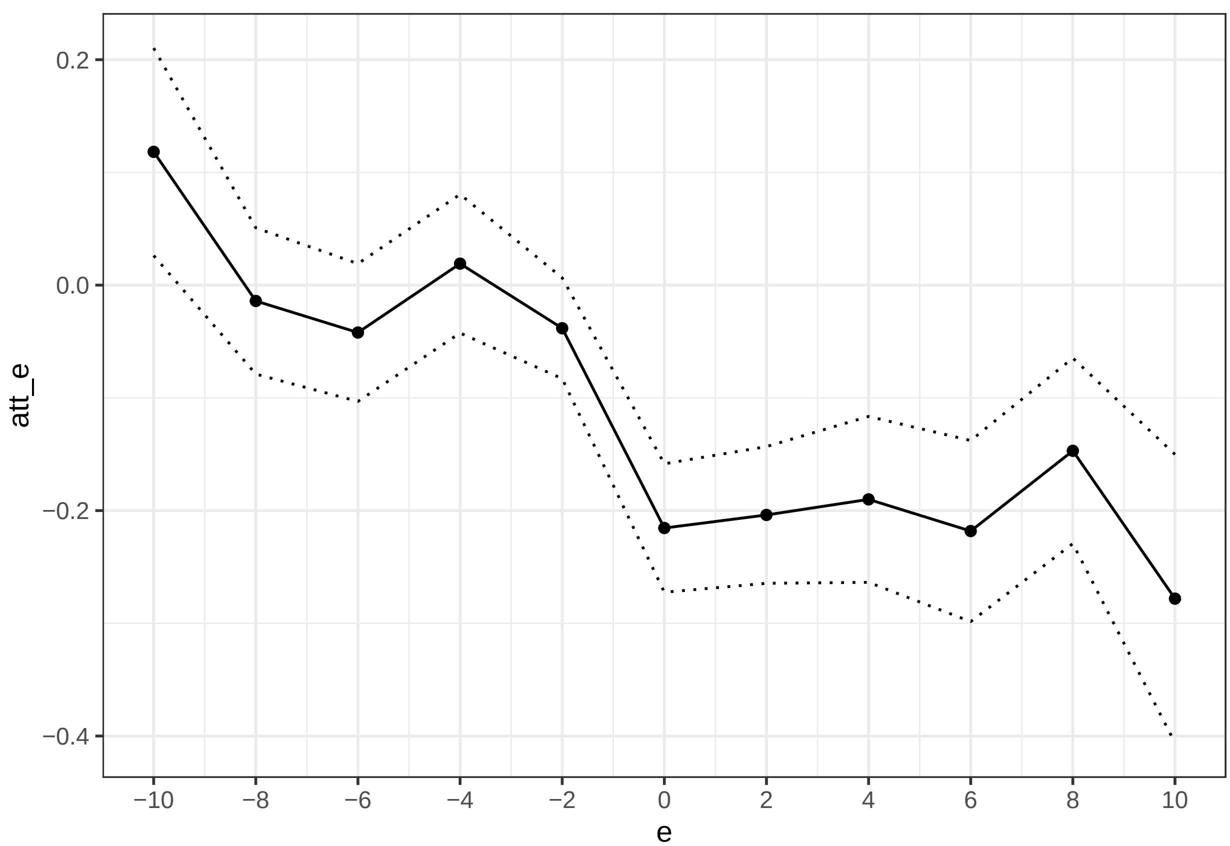
```r
# function to compute event studies
compute.es <- function(attgt_results, w=rep(1,nrow(data))) {
  # event study weights
  eseq <- sort(unique(attgt_results$t - attgt_results$g))
  es_res <- list()
  counter <- 1
  for (e in eseq) {
    this_keepers <- which( (attgt_results$t - attgt_results$g) == e)
    this_attgt <- attgt_results$attgt[this_keepers]
    pg <- sapply(attgt_results$g[this_keepers],
                 function(g)  weighted.mean(data$G==g, w=w))
    pg <- pg / sum(pg)
    att_e <- sum(this_attgt*pg)
    es_res[[counter]] <- c(att_e=att_e, e=e)
    counter <- counter+1
  }
  # convert to data frame
  es_results <- as.data.frame(do.call("rbind", es_res))
  es_results
}

es_results <- compute.es(results)

# bootstrap event study
B <- 100
id_list <- unique(data$id)
boot_es <- list()
for (b in 1:B) {
  boot_weights <- as.numeric(rmultinom(n=1, size=n, prob=rep(1/n,n)))
  this_boot_weights_id <- cbind.data.frame(id=id_list, boot_weights=boot_weights)
  boot_data <- merge(data, this_boot_weights_id, by="id")
  boot_attgt <- compute.attgt(data, w=boot_data$boot_weights)
  boot_es[[b]] <- compute.es(boot_attgt, w=boot_data$boot_weights)$att_e
}
```

```
boot_es <- do.call("rbind", boot_es)
se <- apply(boot_es, 2, sd)
es_results$se <- se
es_results$ciL <- es_results$att_e - 1.96*es_results$se
es_results$ciU <- es_results$att_e + 1.96*es_results$se

library(ggplot2)
ggplot(data=es_results, mapping=aes(x=e,y=att_e)) +
  geom_line() +
  geom_point(size=1.5) +
  geom_line(aes(y=ciU), linetype="dotted") +
  geom_line(aes(y=ciL), linetype="dotted") +
  scale_x_continuous(breaks=seq(-5,5), labels=seq(-10,10,2)) +
  theme_bw()
```



The figure suggests that job displacement causes earnings to drop by, on average, about 20% and that this effect is quite persistent; it appears to be roughly the same 10 years following job displacement. If you look at the estimates in pre-treatment periods, with the exception of 10 years before job displacement, the estimates are fairly close to 0 (and not statistically different from 0) suggesting that the parallel trends assumption is likely to be fairly reasonable in this application.