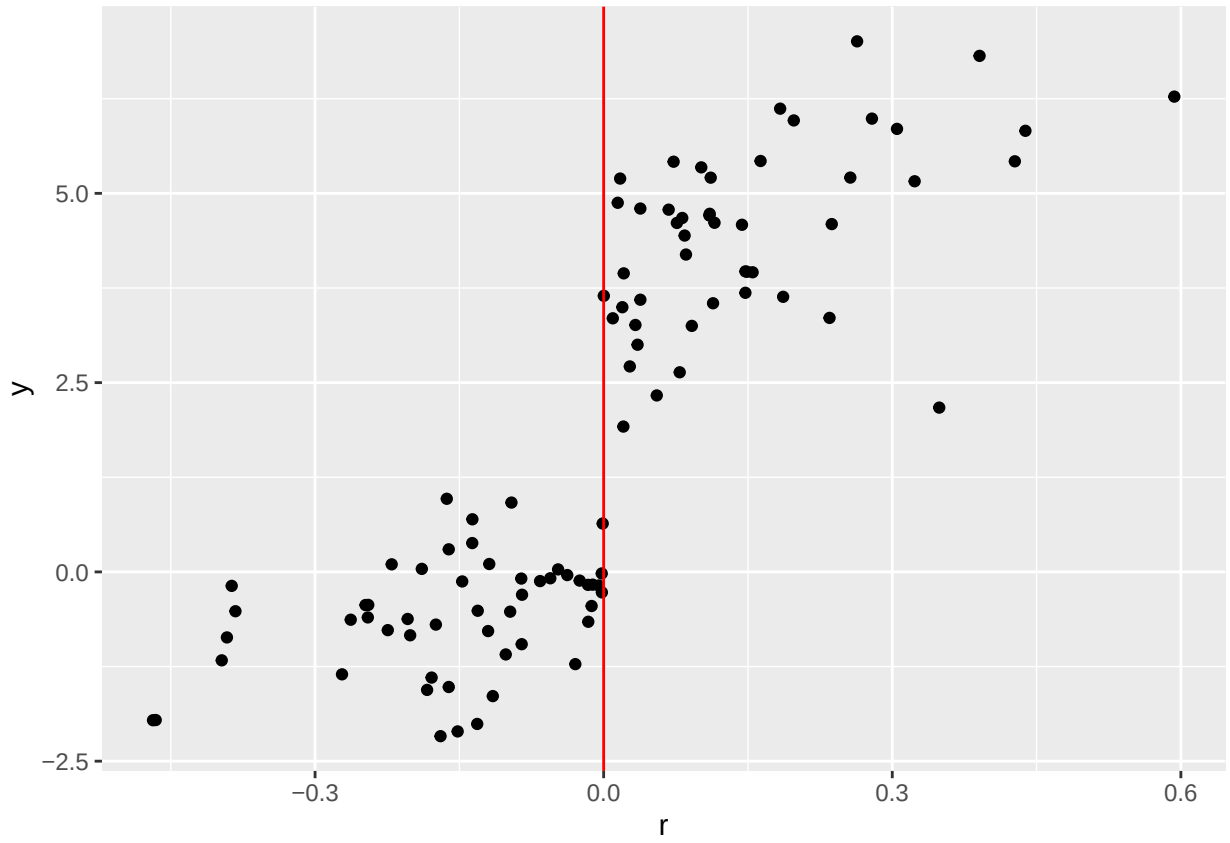
Spatial Regression Discontinuity Designs

Joel Ferguson

11/29/2021

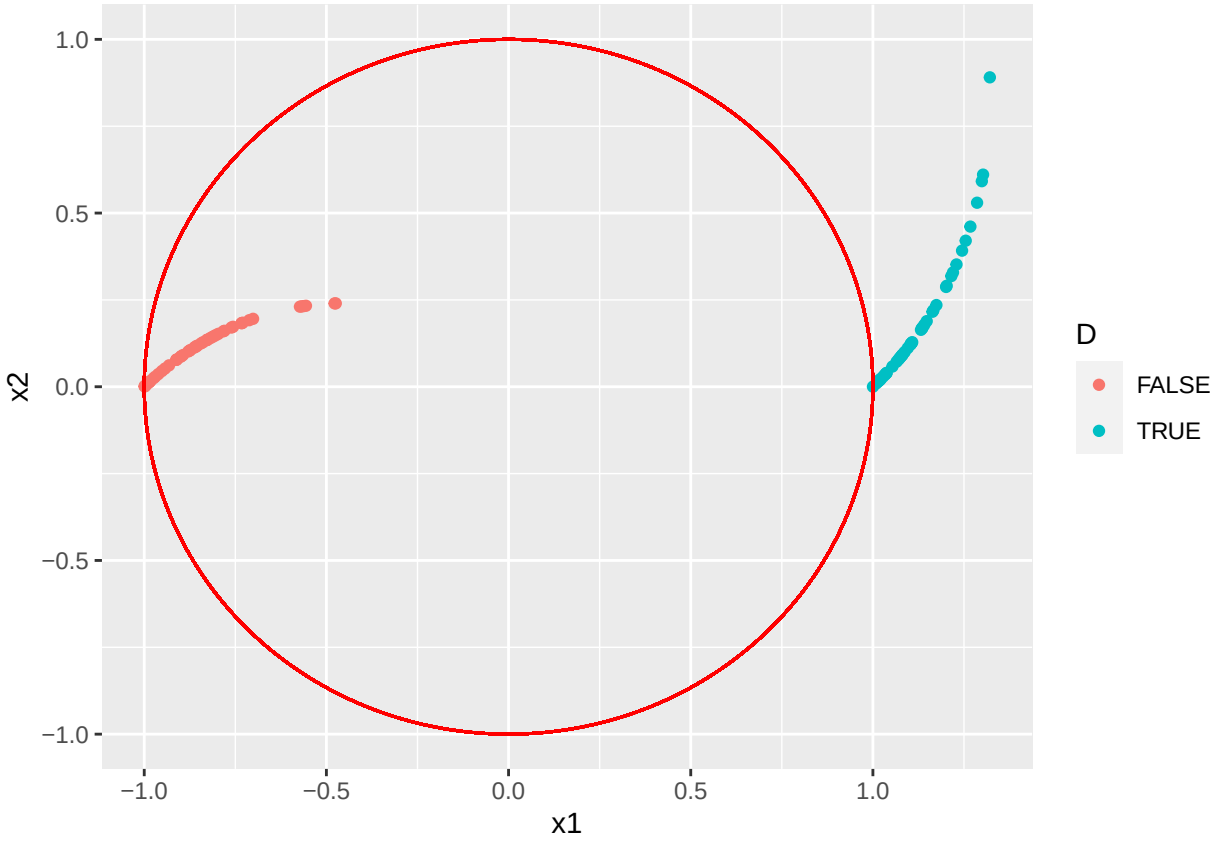
So far, our discussion of RD has been restricted to a one-dimensional running variable. However, a growing number of papers have utilized "boundary" regression discontinuities or higher-dimensional running variables as in Dell & Querubín (2018). Working in a greater number of dimensions than one requires extra attention to be payed to modeling the direct effect of the running variable. To illustrate this, consider the following data:

```
N <- 100
rs <- rnorm(N,0,0.2)
es <- rnorm(N)
y0s <- 4*rs+es
y1s <- y0s+4
Ds <- rs >=0
ys = y1s*Ds+y0s*(1-Ds)

df <- tibble(r=rs,
             D=Ds,
             y=ys)
ggplot(data=df)+
  geom_point(aes(x=r,y=y))+
  geom_vline(aes(xintercept=0),color="red")
```

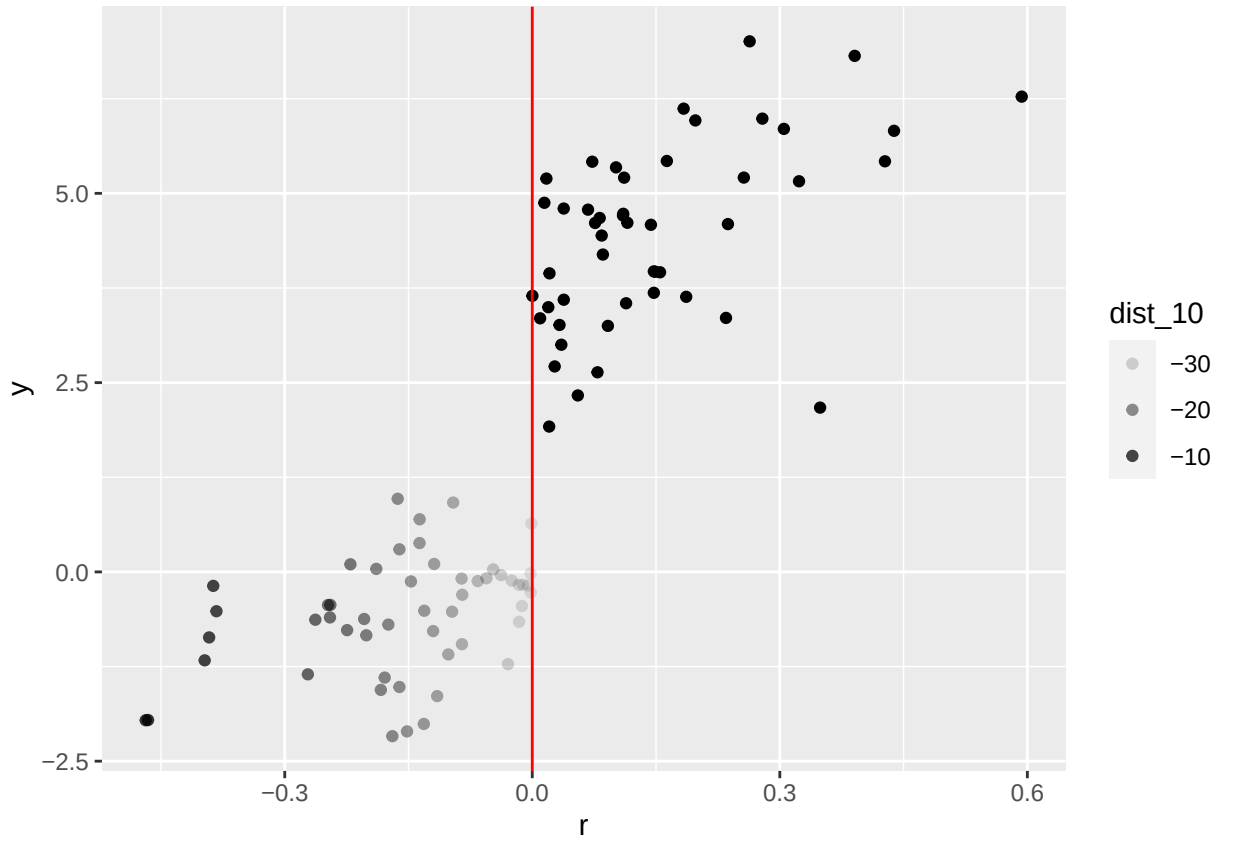


That looks like a pretty clear effect, but you would probably be less convinced of its validity if the points were distributed in space like this:

```
thetas <- (rs<0)*pi+rs
xs <- cos(thetas)+rs*cos(thetas)
ys <- sin(thetas)+rs*sin(thetas)

# check distance to circle is correct
#new_rs <- sqrt((xs-(xs/sqrt(xs^2+ys^2)))^2+(ys-(ys/sqrt(xs^2+ys^2)))^2)

df$x1 <- xs
df$x2 <- ys

ggplot(data=df)+
  geom_point(aes(x=x1,y=x2,color=D))+
  geom_circle(mapping=aes(x0=0,y0=0,r=1),color="red")
```




```
dist_10s <- sqrt((xs-1)^2+ys^2)
df$dist_10 <- -1*(dist_10s^5) # Make the contrast more intense so it's visible

ggplot(data=df)+
  geom_point(aes(x=r,y=y,alpha=dist_10))+
  geom_vline(aes(xintercept=0),color="red")
```



In this case, the untreated units that are farthest from the boundary are actually closest to any treated unit. Seeing the data plotted in space calls into question whether the effect we observe is due to crossing the discontinuity or due to the location of observations. Similar issues arise with multiple disconnected boundaries.

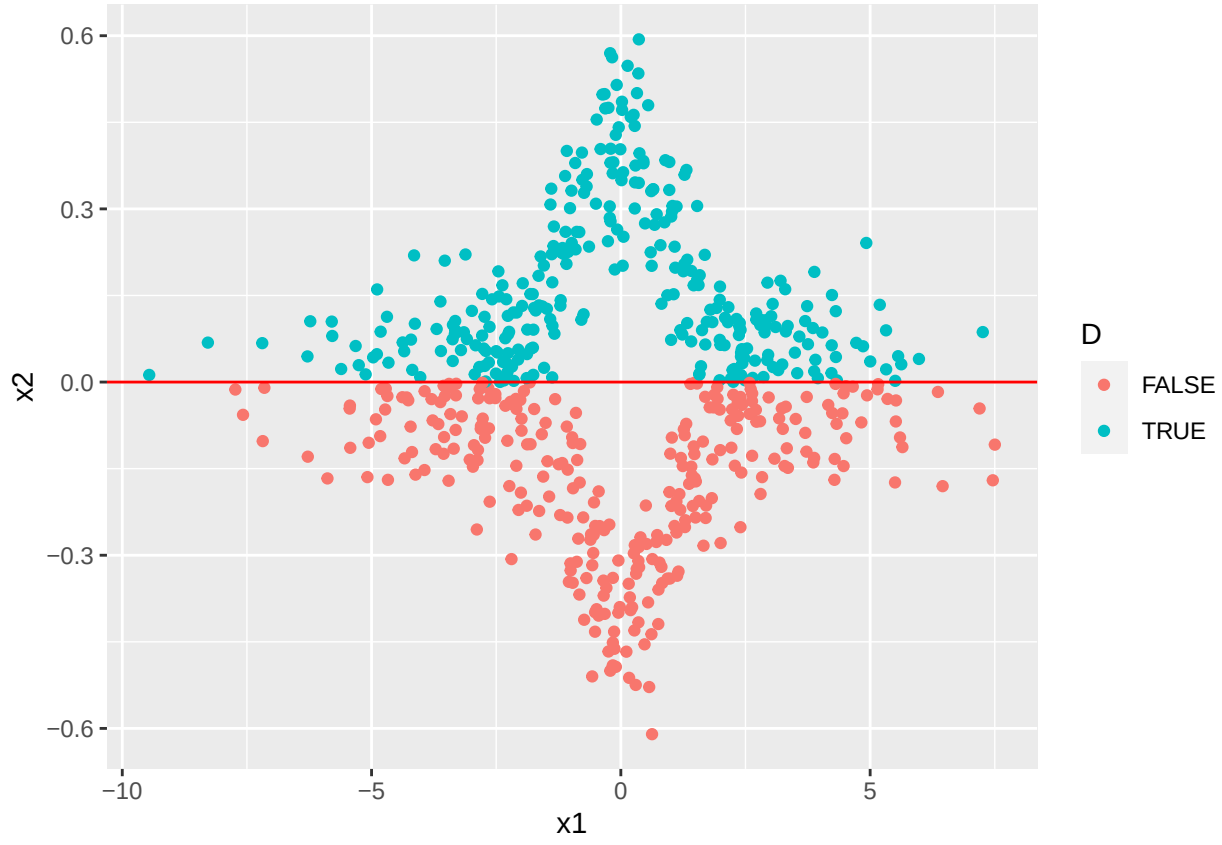
While data are unlikely to be so perversely distributed in real life, it's fairly uncommon to account for this sort of potential issue except for maybe including fixed effects for the particular boundary. As the example above shows, this simple fix may not be enough in certain circumstances. While we probably wouldn't want to run any sort of RD in that example, there are less-extreme scenarios where the units that are closest to the discontinuity are relatively far in space, whereas other units of differing treatment status are close to each other but slightly further from the discontinuity. Papay et al (2011) also highlight that with multi-dimensional discontinuities, we can begin to consider a number of new estimands which may be functionals of the position on the discontinuity.

Imbens and Wager (2019) provide an interesting solution to these problems with many appealing statistical properties. In particular, they seek a linear RD estimator that minimizes MSE in the case where bias is maximized over a class of potential outcome conditional expectation functions with a bounded second derivative at the discontinuity. Finding the estimate requires solving a fairly nasty quadratic programming problem, so I'll just note that you can implement it using their `optrdd` package.

Instead of showing you how to code up that estimator, I'll show you what I *think* is a novel approach to dealing with these issues. The idea is simple, we'll first find a spot along the boundary with a high (distance-weighted) density of nearby observations and then we'll run an ordinary RD with distance to that point as the running variable[1]. This can be extended to finding multiple points along the boundary with high density and then averaging the resulting estimates.

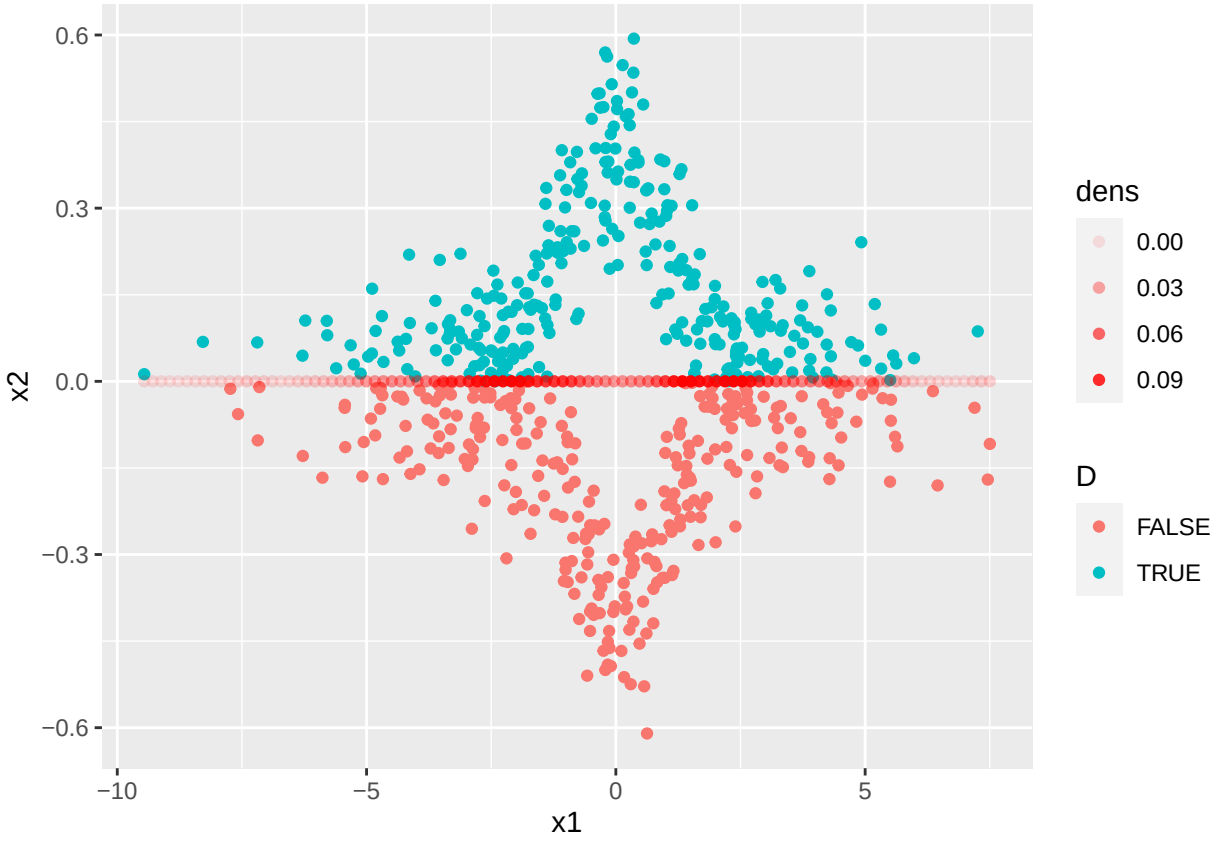
[1] An alternative would be to use a two-dimensional local linear regression to make estimates along the boundary near/at that point

```r
N <- 600
x1s <- rnorm(N,sd=3)
sides <- rbernoulli(N)
x2s <- abs(dnorm(x1s)+rnorm(N,0,0.1))
x2s[sides==0] <- -1*x2s[sides==0]
y0s <- -1*((1/2)^x1s+rnorm(N,0,0.1))
y1s <- x1s^2+rnorm(N,0,0.1)
ys <- sides*y1s+(1-sides)*y0s

df <- tibble(x1=x1s,
             x2=x2s,
             y=ys,
             D=sides)

ggplot(data=df)+
  geom_point(aes(x=x1,y=x2,color=D))+
  geom_hline(aes(yintercept=0),color="red")
```




```r
# Kernel Density Estimation
epi_k <- function(u1,u2,h){ # Epanechnikov Kernel with bandwidth h
  out <- (3/4)*(1-((u1^2+u2^2)/h^2))*as.numeric((sqrt(u1^2+u2^2)/h)<1)
  return(out)
}

k_dens <- function(x1,x2,X1,X2,h){ # Kernel density estimate with data X1,X2
```

```
  f_hat <- 1/(length(X1)*h)*sum(mapply(epi_k, x1-X1, x2-X2, h=h))
  return(f_hat)
}

# Check bandwidth along boundary
dpoints <- seq(from=min(x1s),to=max(x1s),length.out = 100)
densities <- sapply(dpoints,
                    function(x) k_dens(x,0,x1s,x2s,0.5))
dens_df <- tibble(x1=dpoints,
                  dens=densities)

ggplot(data=df)+
  geom_point(aes(x=x1,y=x2,color=D))+
  geom_point(data=dens_df,mapping=aes(x=x1,y=0,alpha=dens),color="red")
```




```
c <- dpoints[which.max(densities)]
c
```

```
## [1] 1.334005
```

```
# Find TE at c
True_TE <- c^2+(1/2)^c
True_TE
```

```
## [1] 2.176236
```

```r
# Filter close to c
df_rd <- df %>%
  mutate(dist=sqrt((x1-c)^2+x2^2)) %>%
  filter(dist <= 0.5)

rd <- feols(y~D*dist,
            data=df_rd)
summary(rd)
```

```
## OLS estimation, Dep. Var.: y
## Observations: 72
## Standard-errors: IID
##              Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -0.265055   0.209802 -1.263357 2.1078e-01
## DTRUE        2.310555   0.339949  6.796772 3.2890e-09 ***
## dist        -0.411622   0.712782 -0.577486 5.6552e-01
## DTRUE:dist  -0.176504   1.091188 -0.161754 8.7198e-01
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.479432   Adj. R2: 0.837965
```

```r
# We can try it again for the next most dense point

c2 <- dpoints[-which.max(densities)][which.max(densities[-which.max(densities)])]
c2
```

```
## [1] 2.361936
```

```r
# Find TE at c
True_TE <- c2^2+(1/2)^c2
True_TE
```

```
## [1] 5.77327
```

```r
# Filter close to c or c2
df_rd2 <- df %>%
  mutate(dist1=sqrt((x1-c)^2+x2^2),
         close1 = dist1<=0.5,
         dist2=sqrt((x1-c2)^2+x2^2),
         close2 = dist2 <= 0.5,
         close=case_when(close1~1,
                         close2~2,
                         T~0),
         dist=pmin(dist1,dist2)) %>%
  filter(close1|close2)

rd2 <- feols(y~D*dist|
               close,
             data=df_rd2)
summary(rd2,se="white")
```

```
## OLS estimation, Dep. Var.: y
## Observations: 137
## Fixed-effects: close: 2
## Standard-errors: Heteroskedasticity-robust
##             Estimate Std. Error   t value  Pr(>|t|)
## DTRUE       4.466603   0.437250 10.215215 < 2.2e-16 ***
## dist        0.203851   0.855905  0.238171   0.81212
## DTRUE:dist -1.988998   1.777407 -1.119044   0.26515
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.12614     Adj. R2: 0.796952
##                 Within R2: 0.752956
```

```
# Let's compare to a simple distance to border RD

df_rd3 <- df %>%
  mutate(dist=abs(x2)) %>%
  filter(dist <= 0.5)

rd3 <- feols(y~D*dist,
             data=df_rd3)
summary(rd3)
```

```
## OLS estimation, Dep. Var.: y
## Observations: 587
## Standard-errors: IID
##             Estimate Std. Error  t value   Pr(>|t|)
## (Intercept) -11.8193    1.56706 -7.54236 1.7799e-13 ***
## DTRUE        25.2262    2.18582 11.54087  < 2.2e-16 ***
## dist         31.7994    7.67207  4.14483 3.9053e-05 ***
## DTRUE:dist  -68.1880   10.76687 -6.33313 4.8023e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 16.6   Adj. R2: 0.20178
```

As you can see, when we start dealing with spatial data in an RD design, we have to be very explicit about what our estimand is and how we're modeling the interaction of where observations are in space with crossing the discontinuity.