Lecture 22

11-12-2021

Announcements

There is no section on 11/24

There is no class on 11/29

All outstanding Checkpoints, WP, and PS3 are due next Monday 11/15

RDD Estimation

RDD estimation presents a variety of design choices

Our discussion is based on the continuity based RD framework

We are interested in "local" or "non-parametric" fits as opposed to global estimates

What is the local polynomial approach?

Recall that there are (generally) no observations for which the score function exactly equals the cutoff value.

We use the fact that any sufficiently smooth function is well approximated by some polynomial function up to some small error term

We will use this point to approximate the regression function near the cutoff

Local Polynomial Point Estimation

Choose a polynomial order *p* and a kernel function $K()$

Choose a bandwidth *h*

Above cutoff fit a WLS regression of the outcome on a constant and weighted coefficients.

$$\hat{\mu_+} + \hat{u_{+,1}}(X_i - c) + \ldots + \hat{u_{+,1}}(X_i - c)^p$$

The result of the intercept from this regression is an estimate of the $\mu_+ = E[Y_i(1)|X_i = c]$

Local Polynomial Point Estimation

For observations below the cutoff we do the same thing but our estimate of interest is the estimated intercept of the local weighted regression $\mu_{-} = E[Y_i(1)|X_i = c]$

Finally, we simply subtract one number from another to get our Sharp RD point estimate

$$t\hat{a}u = \mu_{+} - \mu_{-}$$

Thus there are three design choices: the kernel function, the polynomial order, and the bandwidth

Choosing the Kernel Function

A kernel function assigns non-negative weights to each transformed observation $\frac{X_i - c}{h}$ based on the distance between the observation's score and the cutoff.

Cattaneo *et al* recommend using the triangular kernel function because it has optimal properties for the point estimation.

We also fit a uniform kernel because a uniform kernel is equivalent to simple OLS without weights using only observations within the bandwidth

Local polynomial order

Local polynomial order will determine how the line looks in the estimation function.

A zero order polynomial should not be fit because it has bad technical properties. Higher order polynomials lead to concerns about overfitting.

For a given approximation increasing the polynomial order increases the accuracy but also increases variability

Local linear estimators as implemented by `rdrobust` provide the best trade-off

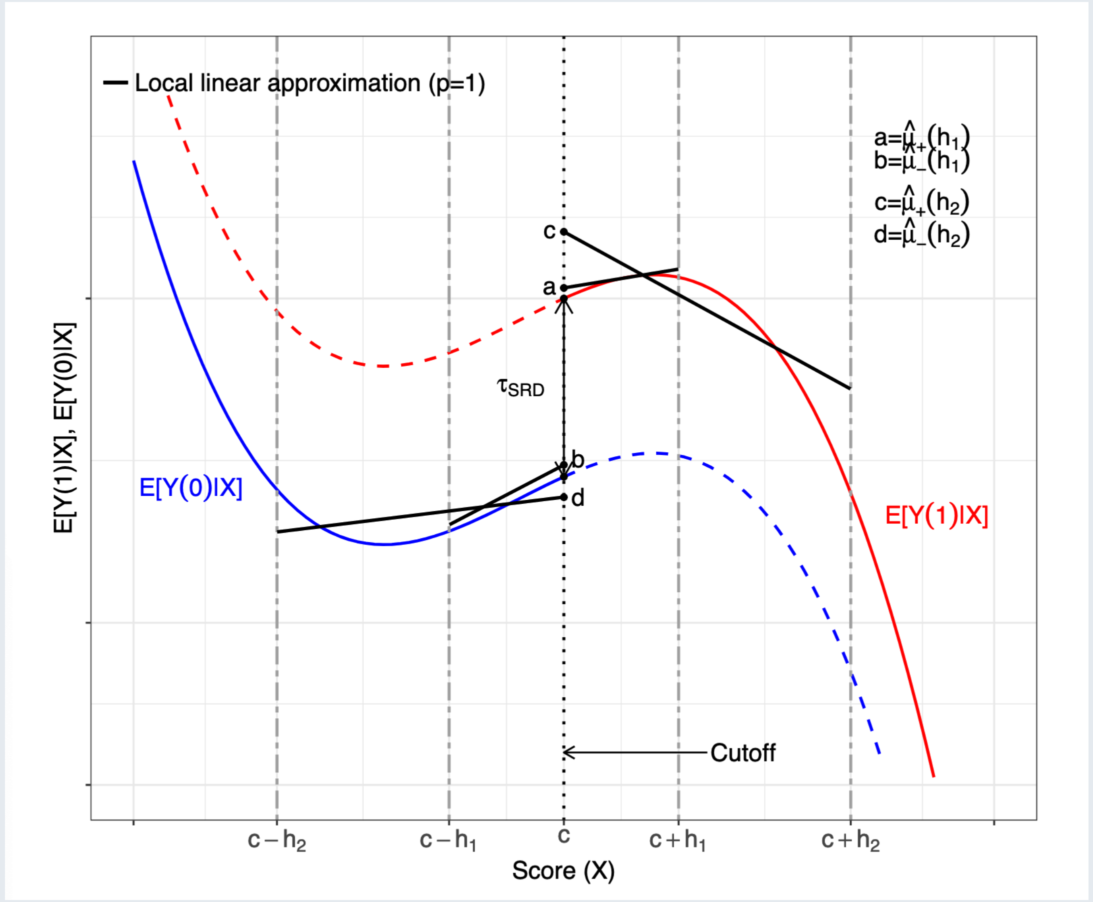
Bandwidth selection

The bandwidth controls the width of the neighborhood around the cutoff that we use to fit our local polynomials to approximate an unknown regression function.

Arguably this is the most consequential decision for an RD design.

Choosing a smaller bandwidth reduces misspecification error but increases variance of the coefficients. Why?

Bias in Local Approximations



Bandwidth Selection

How should we select *h*? The current practice is to balance the bias-variance tradeoff by minimizing the MSE of the local polynomial RD point estimator

$$MSE_\tau = \mathbf{B}^2 + V = h^{2(p+1)}B + \frac{V}{nh}$$

We choose a bandwidth that minimizes that function. Refer to Cattaneo *et al* for the formula. `rdrobust` does this for us which is why we're teaching it.

Point Estimation in Practice

Suppose we have a dataset with an outcome *Y* and a forcing variable *X*

```
c <- 0 # normalized cut off
h <- val # some bandwidth value
low <- lm(Y[X < c+h & X >= c - h] ~ X[X < c+h & X>= c + h])
left_intercept <- left$coefficients[1]

upp <- lm(Y[X > c+h & X =< c - h] ~ X[X > c+h & X <= c + h])
right_intercept <- upp$coefficients[1]
tau <- right_intercept - left_intercept
```

Point Estimation in Practice

We can actually get the same point estimator be fitting a single linear regression that includes an interaction between treatment and score.

These are algebraically equivalent

```
T_X = X*Z
m <- lm(Y[X >= c-h & X <= c + h] ~ X[X >= c-h & X <= c + h]) +
  Z[X >= c-h & X <= c + h] + T_X[X >= c - h & X <= c + h]
summary(m)
```

Point Estimation in Practice

What about a triangular weighting scheme as recommended?

```
w <- NA
w[X < c- h & X >= -(c-h)] = 1 - abs(X[X < c-h & X >=-(c-h)]/(c+h))
w[X >= c- h & X <= (c+h)] = 1 - abs(X[X >= c-h & X <= c+h]/(c+h))

## Put them into OLS
m2 <- lm(Y[X<c-h]~X[X < c-h], weights = w[X < c-h])
m3 <- lm(Y[X >= c-h]~X[X>=c-h], weights = w[X>=c-h])
tau <- m3$coefficients[1] - m2$coefficients[1]
```

Point Estimation in Practice

```r
out <- rdrobust(Y, X, kernel = "triangular", p = 1, bwselect = "mserd")
summary(out)
```

What about covariates

Our approach here is non-parametric. That means that in order to be consistent with this approach our covariates must be balanced at the cutoff

Covariate adjustment will not restore identification of the standard RD design when treated and control units differ systematically at the cutoff

This is the empirical test that previous papers we have discussed (e.g. Mello 2019) failed when considering the RD specification

With appropriate covariates, our estimates will have increased precision.

What about covariates

```r
# Assume C is a column matrix of covariates
out_c <- rdrobust(Y, X, covs = C,
                  kernel = "triangular", p = 1,
                  bwselect = "mserd")

# Will clustering
out_cl <- rdrobust(Y, X, covs = C,
                   kernel = "triangular", p = 1,
                   bwselect = "mserd", cluster = cluster_var)
```