Regression discontinuity design

Tutorial 6

Stanislav Avdeev

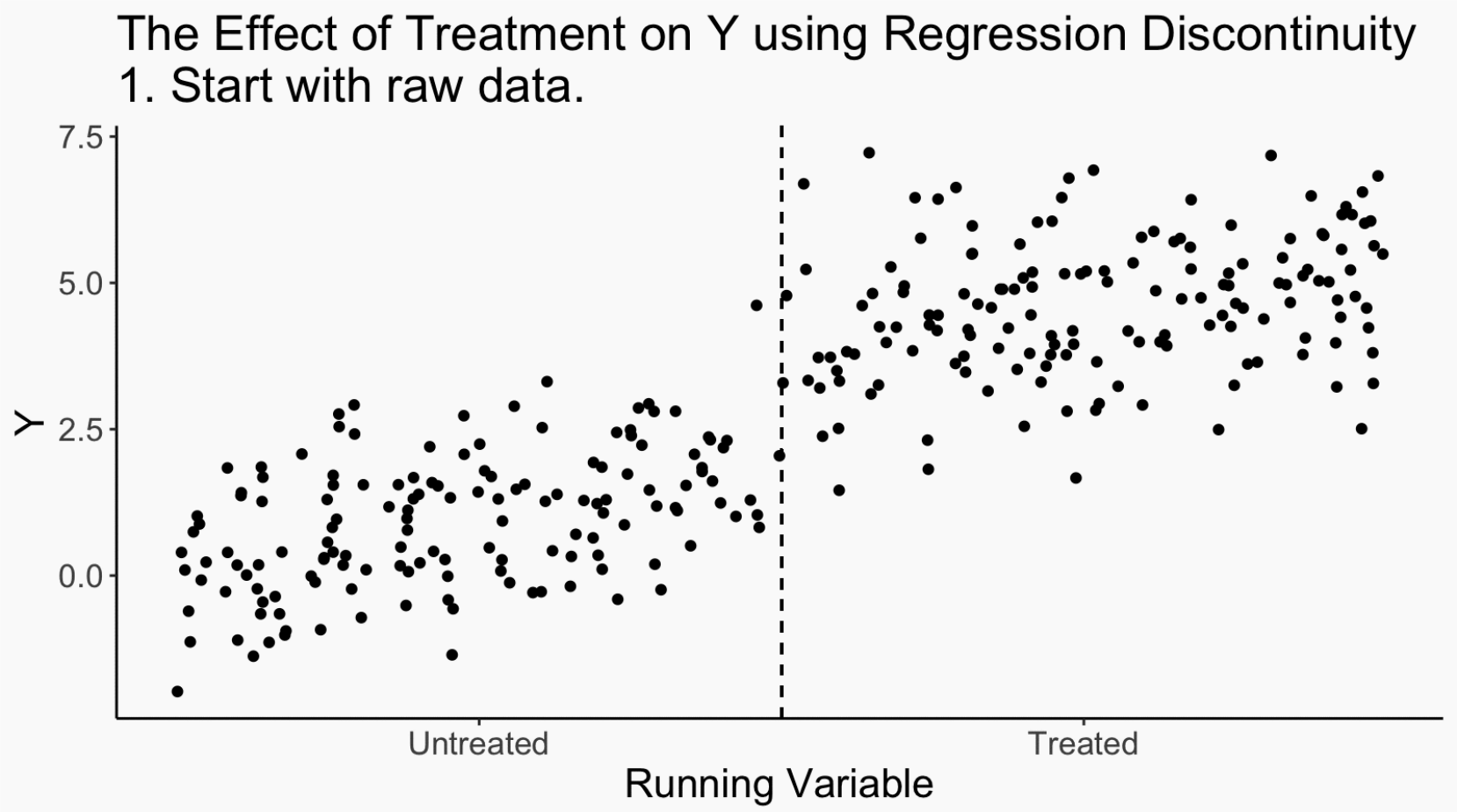
Goal for today's tutorial

1. Discuss (fuzzy) regression discontinuity design - RDD
2. Discuss regression kink design - RKD
3. Discuss functional form, bandwidth, and controls in RDD
4. Check assumptions of RDD
5. Discuss sensitivity checks

RDD

- The basic idea of RDD is to look for a treatment that is assigned on the basis of being above/below a **cutoff value** of a continuous variable, for example
  - if a candidate gets 50.1% of the vote they're in, 40.9% and they're out
  - if you're 65.1 years old you get Medicaid, if you're 64.9 years old you don't
  - if you score above 75, you'll be admitted into a "gifted and talented" (GATE) program
- We call these continuous variables **running variables** because we run along them until we hit the cutoff
- Basically, the idea is that right around the cutoff, treatment is **randomly assigned**
  - if you have a test score of 74.9 (not high enough for GATE), you're basically the same as someone who has a test score of 75.1 (just barely high enough)
- So we have two groups - the just-barely-missed-outs and the just-barely-made-its - that are basically exactly the same except that one happened to get treatment
  - this gives us the effect of treatment **for people who are right around the cutoff** - LATE

RDD: graphically



RDD: same slope

- The most basic version of RDD allows for a jump but forces the slope to be the **same** on either side

$$Y_i = \beta_0 + \beta_1 Treated_i + \beta_2 XC_i + U_i$$

  where
  - $Treated_i$ is a binary variable equal to 1 if you are above the cutoff
  - XC_i is the running variable that is centered around the cutoff, i.e. $XC_i = X_i - Cutoff$
  - β_1 is how the intercept jumps - that is the RDD effect
- Remember that the RDD estimates the average treatment effect among those just **around the cutoff** - LATE

RDD: simulation

- Let us simulate a dataset with the same slope

```
set.seed(7)
df ← tibble(X = runif(500),
            Treated = ifelse(X > 0.5, 1, 0),
            XC = X - 0.5,
            Y = 0.7*Treated + XC + rnorm(500, 0, 0.3))
```

X	Treated	XC	Y
0.4985591	0	-0.0014409	-0.2345307
0.4994985	0	-0.0005015	-0.2821893
0.5016948	1	0.0016948	0.8777840
0.5027101	1	0.0027101	0.5410091

RDD: simulation

```
# The true effect is 0.7
m ← lm(Y ~ Treated + XC, df)
```

	Model 1
(Intercept)	−0.032
	(0.029)
Treated	0.765***
	(0.053)
XC	0.930***
	(0.095)
Num.Obs.	500
+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001	

RDD: varying slopes

- Typically, you will want to let the slope **vary** to either side
- In effect, we are fitting an entirely different regression line on each side of the cutoff
- We can do this by using the following regression

$$Y_i = \beta_0 + \beta_1 Treated_i + \beta_2 XC_i + \beta_3 Treated_i \times XC_i + U_i$$

where

- β_1 is how the intercept jumps - that's the RDD effect
- β_3 is how the slope changes - that's the RKD effect

RDD: simulation

Let us simulate a dataset with the same slope

```
set.seed(7)
df ← tibble(X = runif(500),
            Treated = ifelse(X > 0.5, 1, 0),
            XC = X - 0.5,
            Y = 0.7*Treated + XC + 0.5*Treated*XC + rnorm(500, 0, 0.3))
```

X	Treated	XC	Y
0.4985591	0	-0.0014409	-0.2345307
0.4994985	0	-0.0005015	-0.2821893
0.5016948	1	0.0016948	0.8786314
0.5027101	1	0.0027101	0.5423641

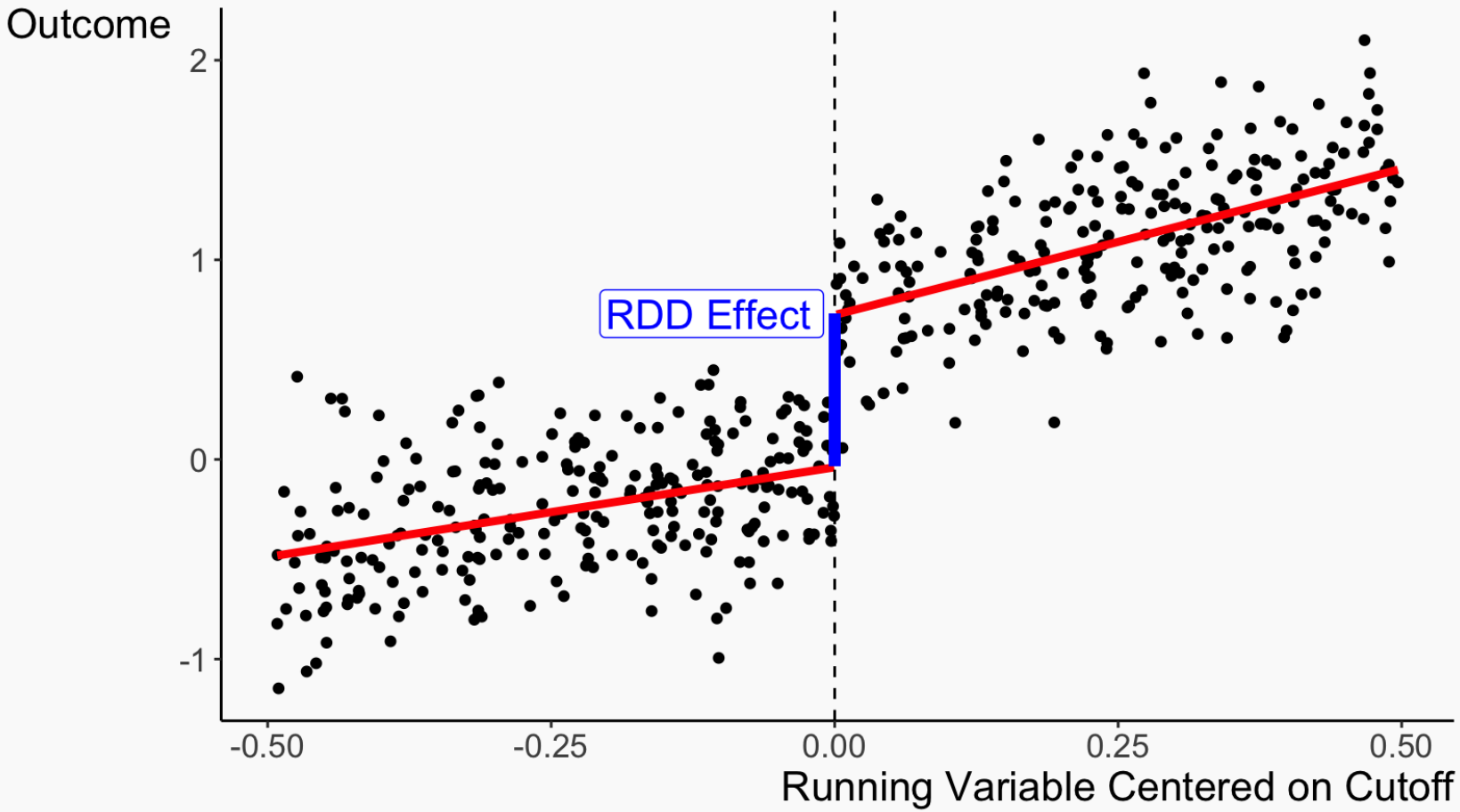
RDD: simulation

```
# The true RDD effect is 0.7, and the true RKD effect is 0.5
m ← lm(Y ~ Treated*XC, df)
```

	Model 1
(Intercept)	−0.039
	(0.035)
Treated	0.763***
	(0.054)
XC	0.899***
	(0.129)
Treated × XC	0.565**
	(0.190)
Num.Obs.	500
+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001	

RDD: graphically

- The true model is an RDD effect of 0.7 with a slope of 1 to the left of the cutoff and a slope of 1.5 to the right, so the RKD effect is 0.5



Choices

- Bandwidth
- Functional form
- Controls

Choices: bandwidth

- The idea of RDD is that people **just around the cutoff** are very much comparable
- So people far away from the cutoff are not too informative
  - at best they help determine the slope of the fitted lines
- So we might limit our analysis within just a **narrow window** around the cutoff
- This makes the **exogenous-at-the-jump assumption** more plausible
  - this lets us worry less about functional form over a narrow range, as there is not too much difference between a linear and a square term
  - but it reduces our sample size considerably
- There's a big literature on **optimal bandwidth selection** which balances the addition of bias (from adding people far away from the cutoff) vs. variance (from adding more people so as to improve estimator precision)
- Gelman & Imbens (2019) show that the "naive" RDD estimators place high weights on observations far from the threshold
  - so it's better to drop these observations

Choices: bandwidth

- Pay attention to the accuracy, standard errors, and sample sizes

```
# The true effect is 0.7
m1 ← lm(Y ~ Treated*XC, df)
m2 ← lm(Y ~ Treated*XC, df %>% filter(abs(XC) < 0.25))
m3 ← lm(Y ~ Treated*XC, df %>% filter(abs(XC) < 0.1))
m4 ← lm(Y ~ Treated*XC, df %>% filter(abs(XC) < 0.05))
m5 ← lm(Y ~ Treated*XC, df %>% filter(abs(XC) < 0.01))
```

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**
Treated	0.763***	0.796***	0.730***	0.792***	1.065**
	(0.054)	(0.071)	(0.111)	(0.130)	(0.270)
Num.Obs.	500	259	95	53	19
+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001					

Choices: functional form

- Why do we fit only a straight line on either side?
  - if the true relationship is curvy this will give us the wrong result
- We can be much more flexible, by including polynomials

$$Y_i = \beta_0 + \beta_1 Treated_i + \beta_2 XC_i + \beta_3 Treated_i \times XC_i$$
$$+\beta_4 XC_i^2 + \beta_5 Treated_i \times XC_i^2 + U_i$$

where

  - β_1 remains our jump at the cutoff - the RDD estimate

Choices: functional form

- The interpretation is the same as before - look for the jump
- We want to be careful with polynomials though, and not add too many
  - remember, the more polynomial terms we add, the stranger the behavior of the line at either end of the range of data
  - so we can get illusory effects generated by having too many terms
- A common approach is to use **non-parametric** regression or **local linear regression**
  - this does not impose any particular shape
  - and it's easy to get a prediction on either side of the cutoff
  - this allows for non-straight lines without dealing with polynomials

Choices: functional form

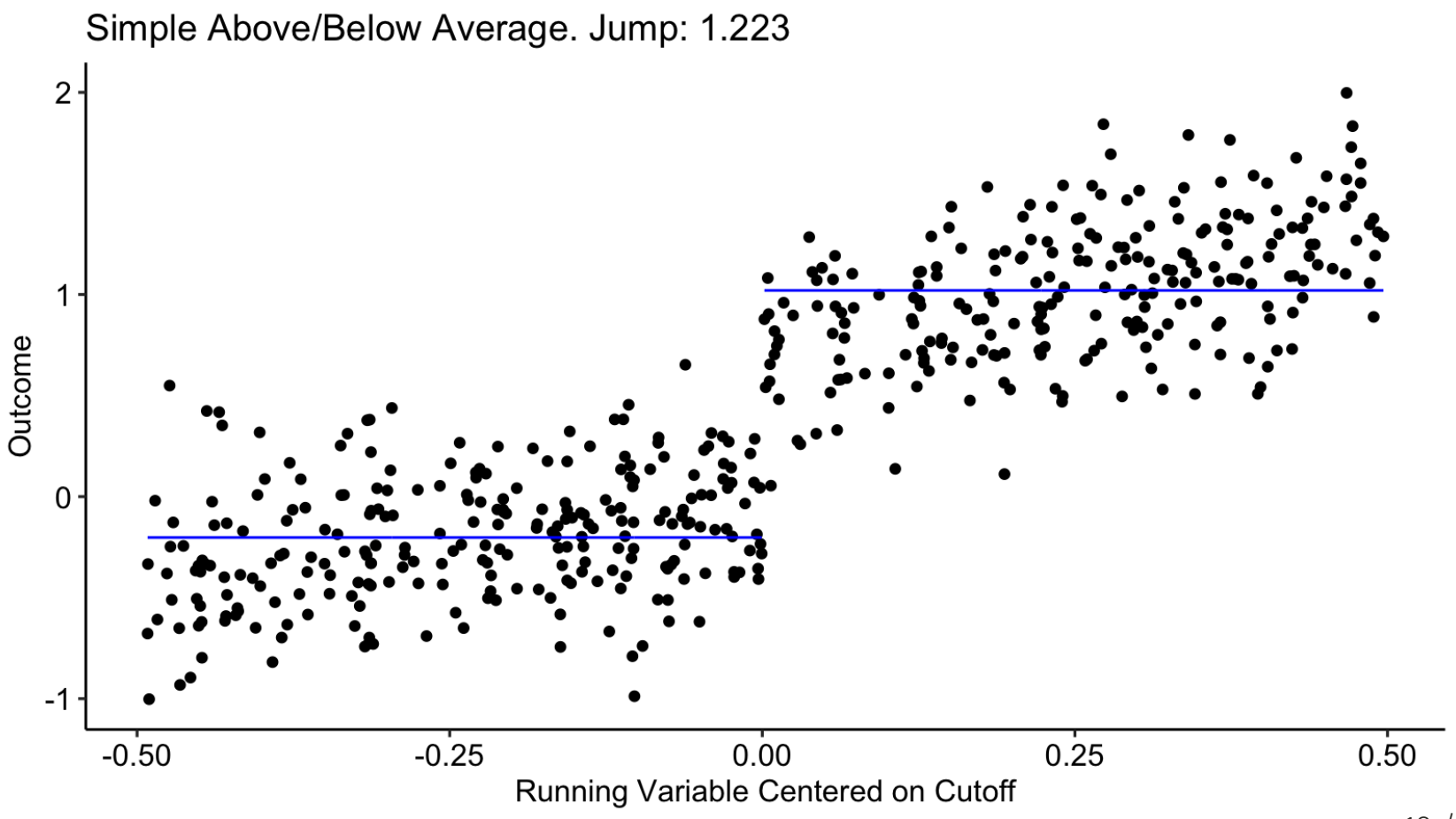
- Let's look at the same data with a few different functional forms

```
set.seed(7)
df ← tibble(X = runif(500),
            Treated = ifelse(X > 0.5, 1, 0),
            XC = X - 0.5,
            Y = 0.7*Treated + XC + 0.6*XC^2 + rnorm(500, 0, 0.3))
```

X	Treated	XC	Y
0.4985591	0	-0.0014409	-0.2345295
0.4994985	0	-0.0005015	-0.2821892
0.5016948	1	0.0016948	0.8777858
0.5027101	1	0.0027101	0.5410135

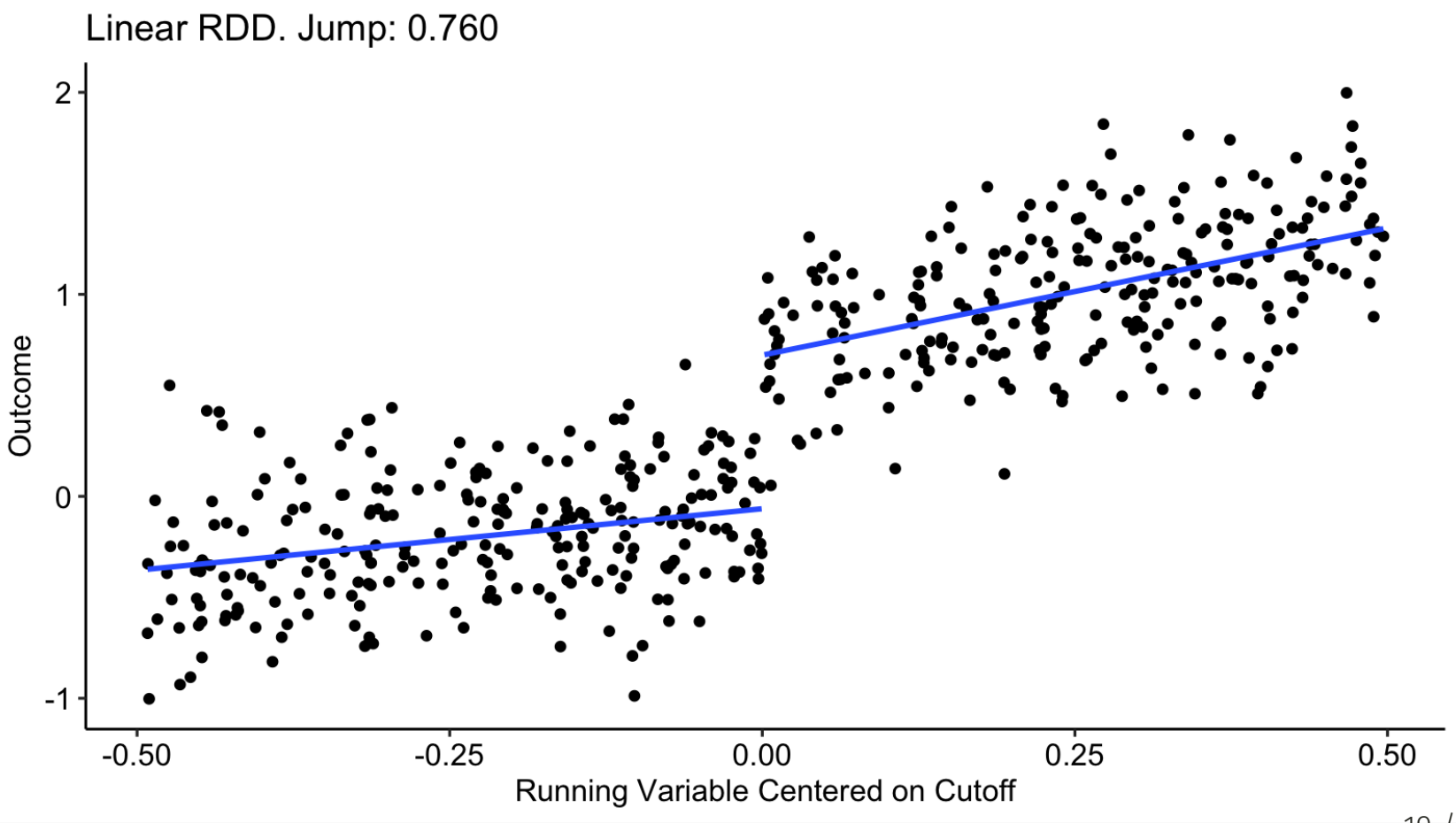
Choices: functional form

```
# The true effect is 0.7, and the true model is an order-2 polynomial
m ← lm(Y ~ Treated, df)
```



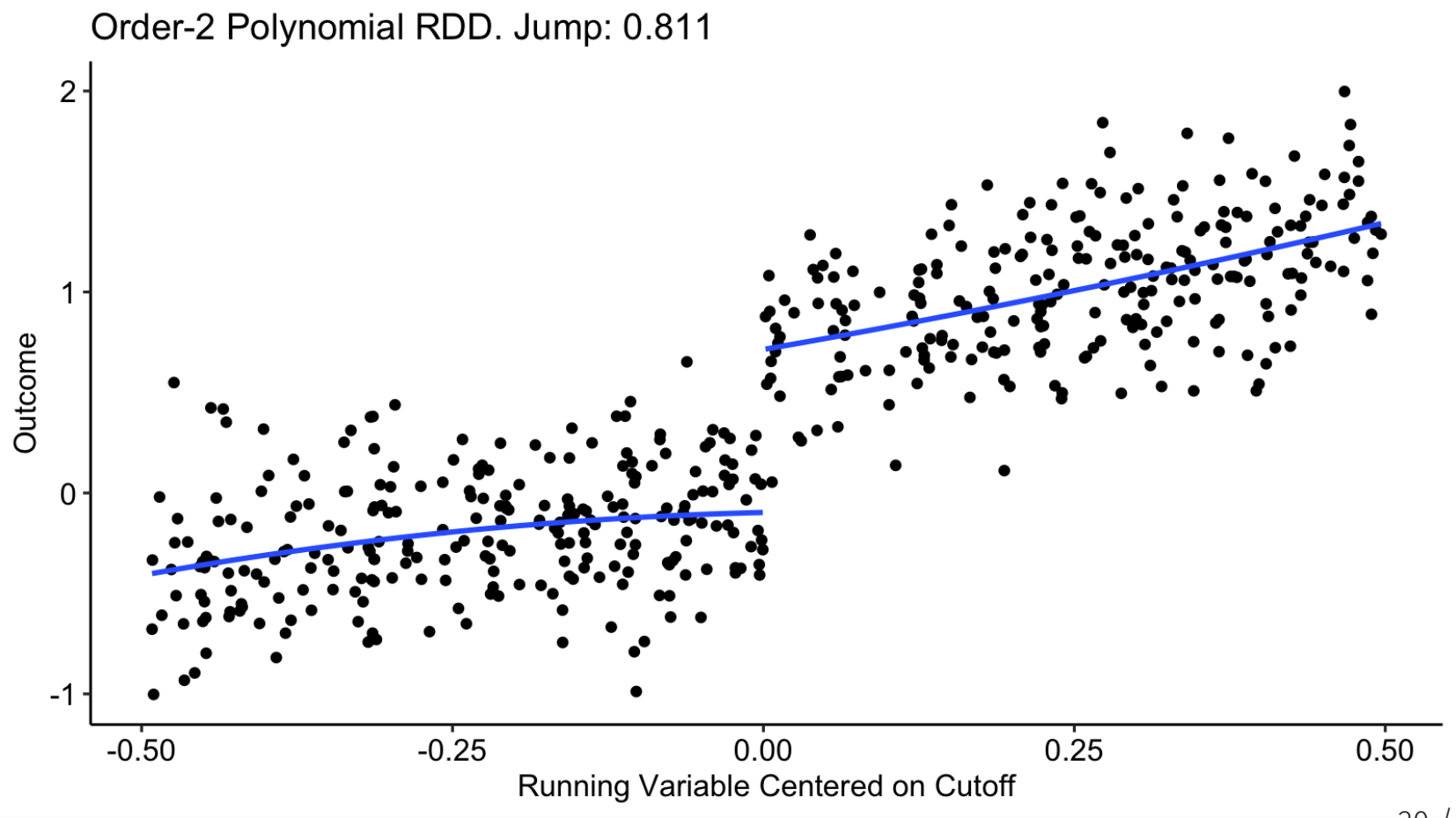
Choices: functional form

```
# The true effect is 0.7, and the true model is an order-2 polynomial
m ← lm(Y ~ Treated*XC, df)
```



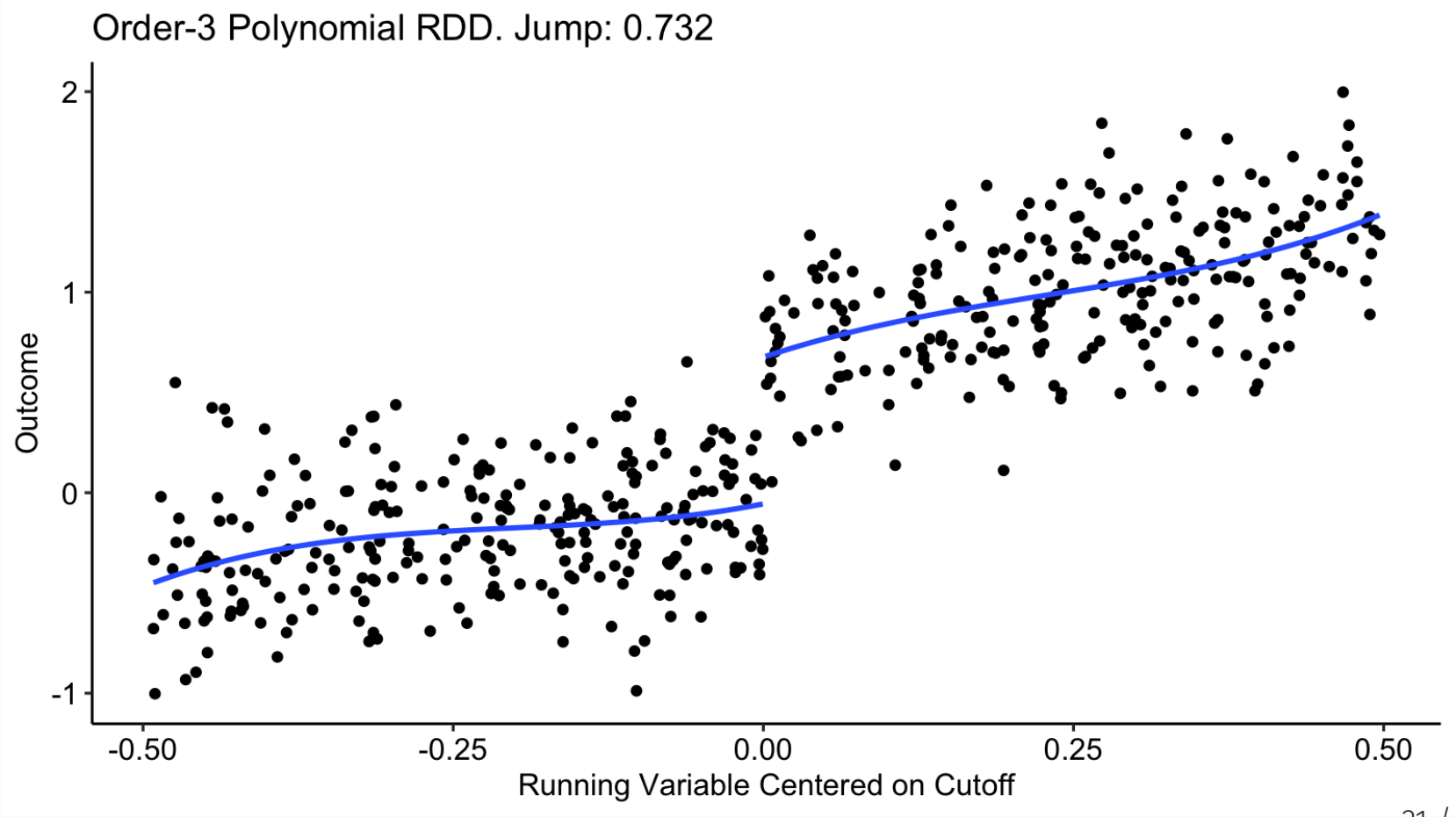
Choices: functional form

```
# The true effect is 0.7, and the true model is an order-2 polynomial
m ← lm(Y ~ Treated*XC + Treated*I(XC^2), df)
```



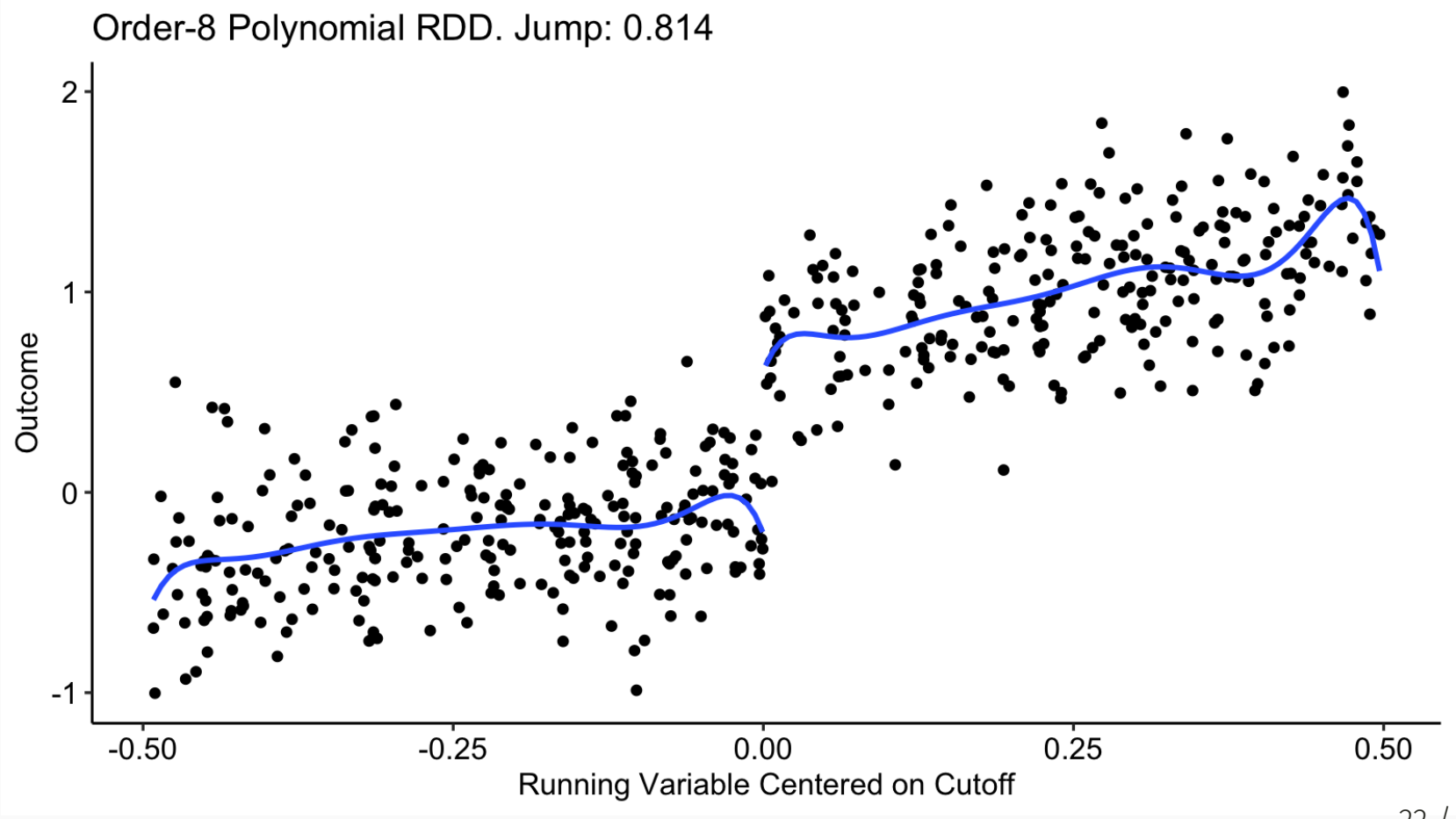
Choices: functional form

```
# The true effect is 0.7, and the true model is an order-2 polynomial
m ← lm(Y ~ Treated*XC + Treated*I(XC^2) + Treated*I(XC^3), df)
```



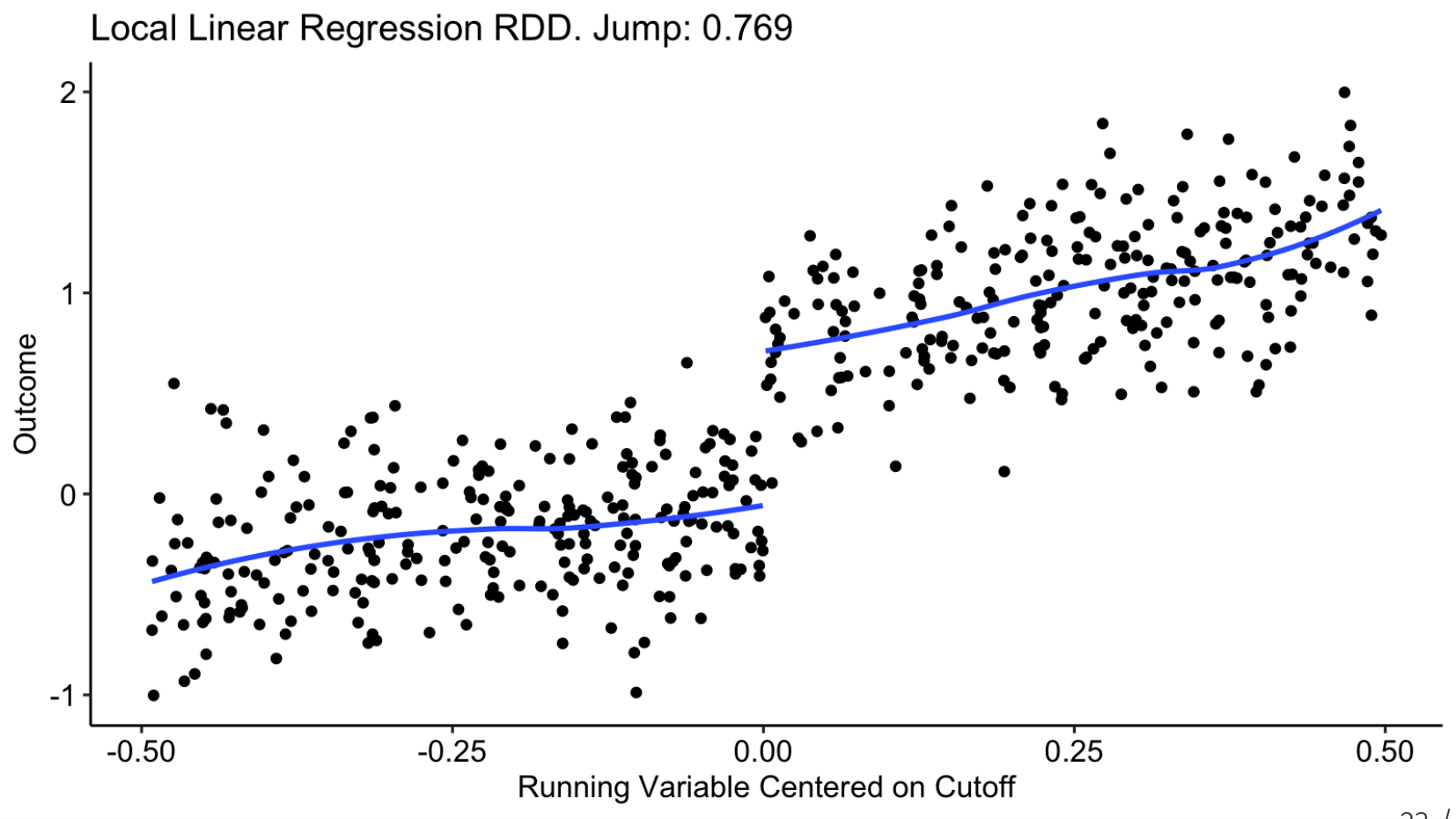
Choices: functional form

```
m ← lm(Y ~ Treated*XC + Treated*I(XC^2) + Treated*I(XC^3) + Treated*I(XC^4) +
       Treated*I(XC^5) + Treated*I(XC^6) + Treated*I(XC^7) + Treated*I(XC^8), df)
```



Choices: functional form

```
# The true effect is 0.7, and the true model is an order-2 polynomial
# The estimated model is recommended by Gelman & Imbens (2019)
```



Choices: functional form

- A conclusion is to **avoid** higher-order polynomials
  - even the true model can be worse than something simpler sometimes
  - and fewer terms makes more sense too, once we apply a bandwidth and zoom in
  - consider a nonparametric approach
- Gelman & Imbens (2019) argue that controlling for global high-order polynomials in RDD has three major problems
  - noisy estimates
  - sensitivity to the degree of the polynomial
  - poor coverage of confidence intervals
- Be very suspicious if your fit is wildly off right around the cutoff

Choices: controls

- Generally you don't need control variables in an RDD
  - if the design is valid, RDD is almost like a randomised experiment
- Although maybe we want some controls if we a bandwidth is **wide**
  - this will remove some of the bias
- Control variables also allow us to perform **placebo tests** of our RDD model
  - we can rerun our RDD model, but simply use a **control** variable as the **outcome**
  - we should not find any effect
  - you can run these for every control variable you have

Assumptions

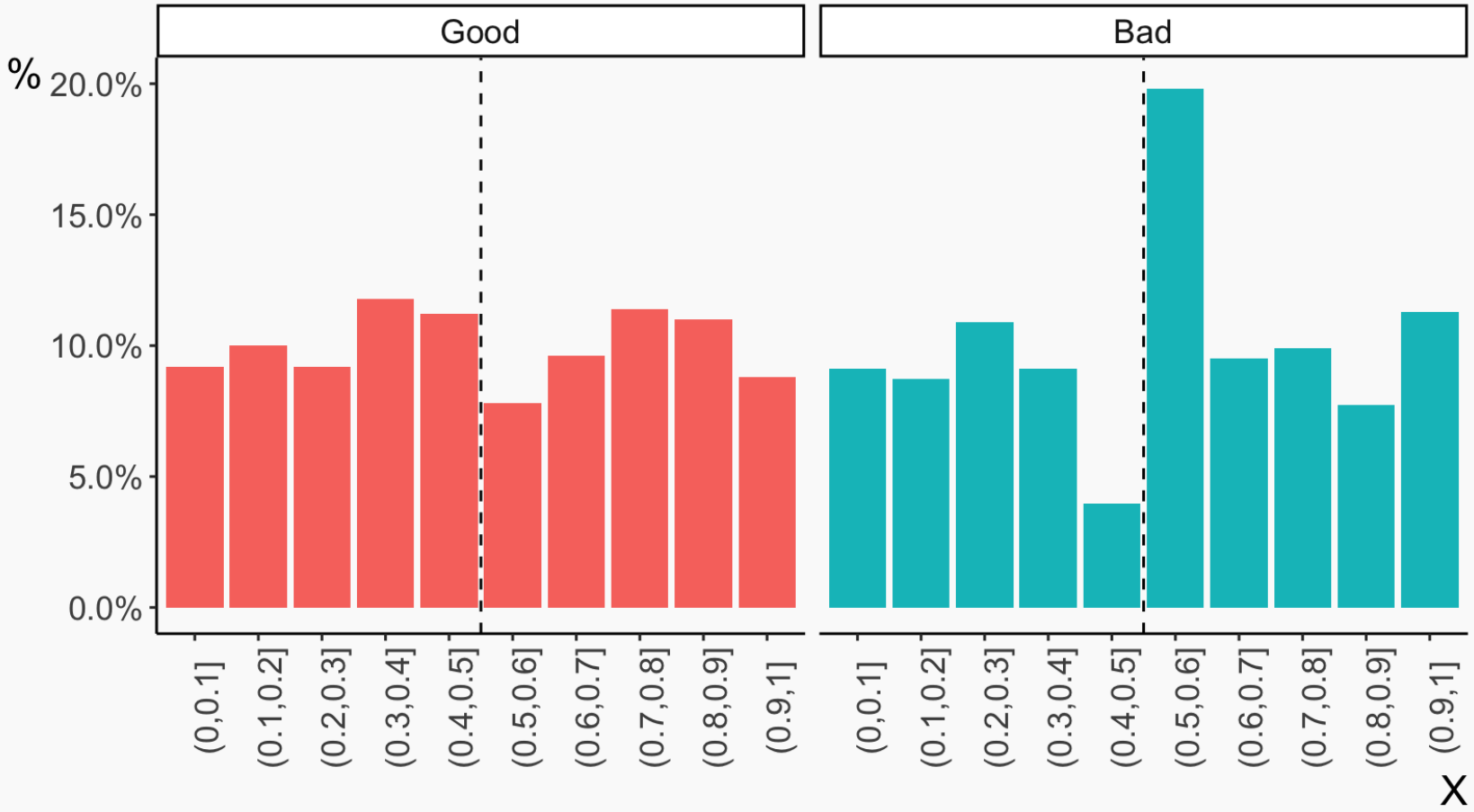
- We knew there must be some assumptions
  - some are more obvious, i.e. we should be using the correct functional form
  - others are trickier, i.e. what are we assuming about the error term and endogeneity?
- Specifically, we are assuming that the only thing jumping at the cutoff is **treatment**
  - sort of like parallel trends, but maybe more believable since we've narrowed in
- For example, if having an income below 150% of the poverty line gets you access to food stamps **and** to job training, then we can't really use that cutoff to get the effect of **just** food stamps
- The only thing different about just above/just below should be treatment
  - but what if the running variable is **manipulated**?

Assumptions: bunching

- Imagine you are a teacher grading the gifted-and-talented exam. You see someone with an 74 and think "they are so close, I'll just give them an extra point"
  - suddenly, that treatment is a lot less randomly assigned around the cutoff
- If there's manipulation of the running variable around the cutoff, we can often see it in the presence of **bunching**
  - in other words, there is a big **cluster of observations** to one side of the cutoff and a seeming gap missing on the other side
- How can we check this?
  - we can look graphically by just checking for a jump at the cutoff in **number of observations** after binning
  - we can use the **McCrary density test** in `rddensity` package

Assumptions: bunching

- The first one looks pretty good. The second one looks not-so-good



Assumptions: bunching

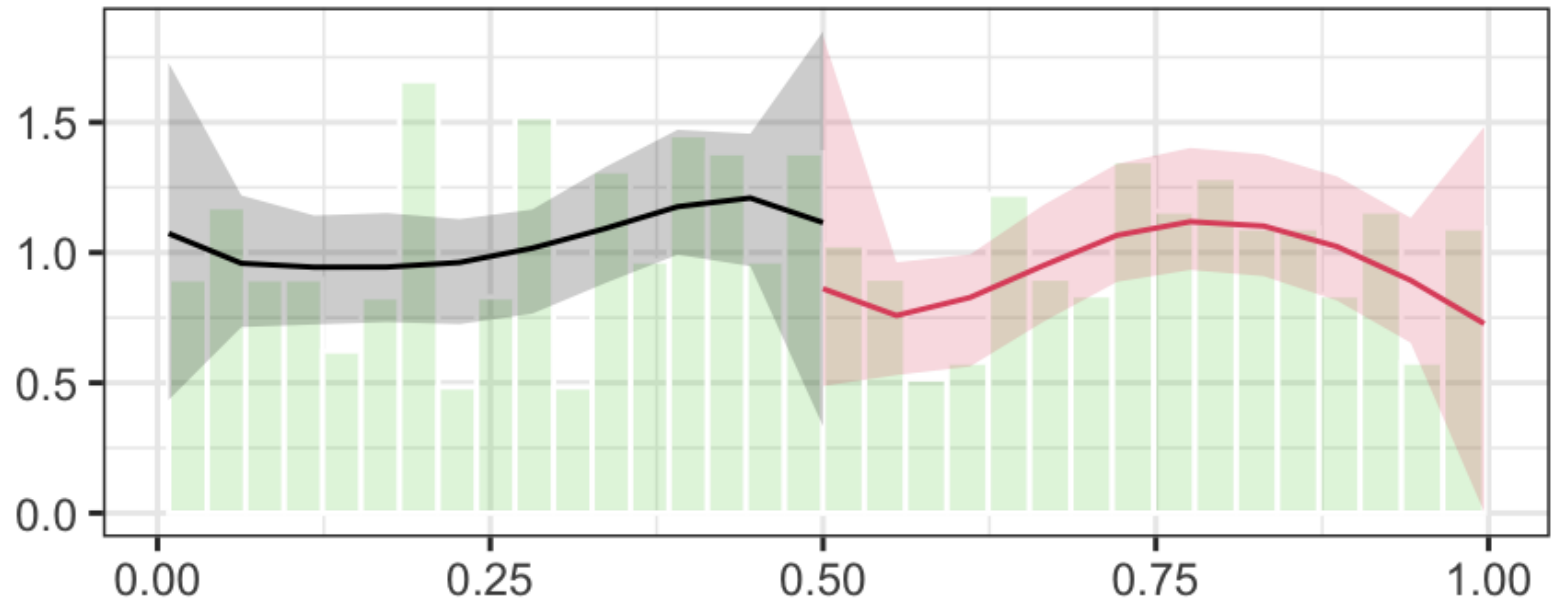
```
library(rddensity)
test_density ← rddensity(df$X, c = 0.5)
```

- The p-value of a t-test 0.8959 shows no manipulation

```
##
## Manipulation testing using local polynomial density estimation.
##
## Number of obs =       500
## Model =               unrestricted
## Kernel =              triangular
## BW method =           estimated
## VCE method =          jackknife
##
## c = 0.5               Left of c           Right of c
## Number of obs         257                 243
## Eff. Number of obs    113                 71
## Order est. (p)        2                   2
## Order bias (q)        3                   3
## BW est. (h)           0.193               0.17
##
## Method                T                   P > |T|
## Robust                0.1309              0.8959
```

Assumptions: bunching

```
plot_density_test ← rdplotdensity(rdd = test_density, X = df$X)
```



- Notice that the confidence intervals overlap substantially

Fuzzy RDD

- What if treatment is not determined sharply by the cutoff?
  - we can account for this with a model designed to take this into account
- Specifically, we can use the IV method
  - basically, IV estimates how much the **chances of treatment** go up at the cutoff, and scales the **estimate of treatment** by that change (remember 4^{th} TA about LATE)
  - we can perform the IV method using `feols()` in `fixest`
- What happens if we just do RDD as normal?
  - the effect is **underestimated** because we have some untreated in the post-cutoff and treated in the pre-cutoff

Fuzzy RDD: simulation

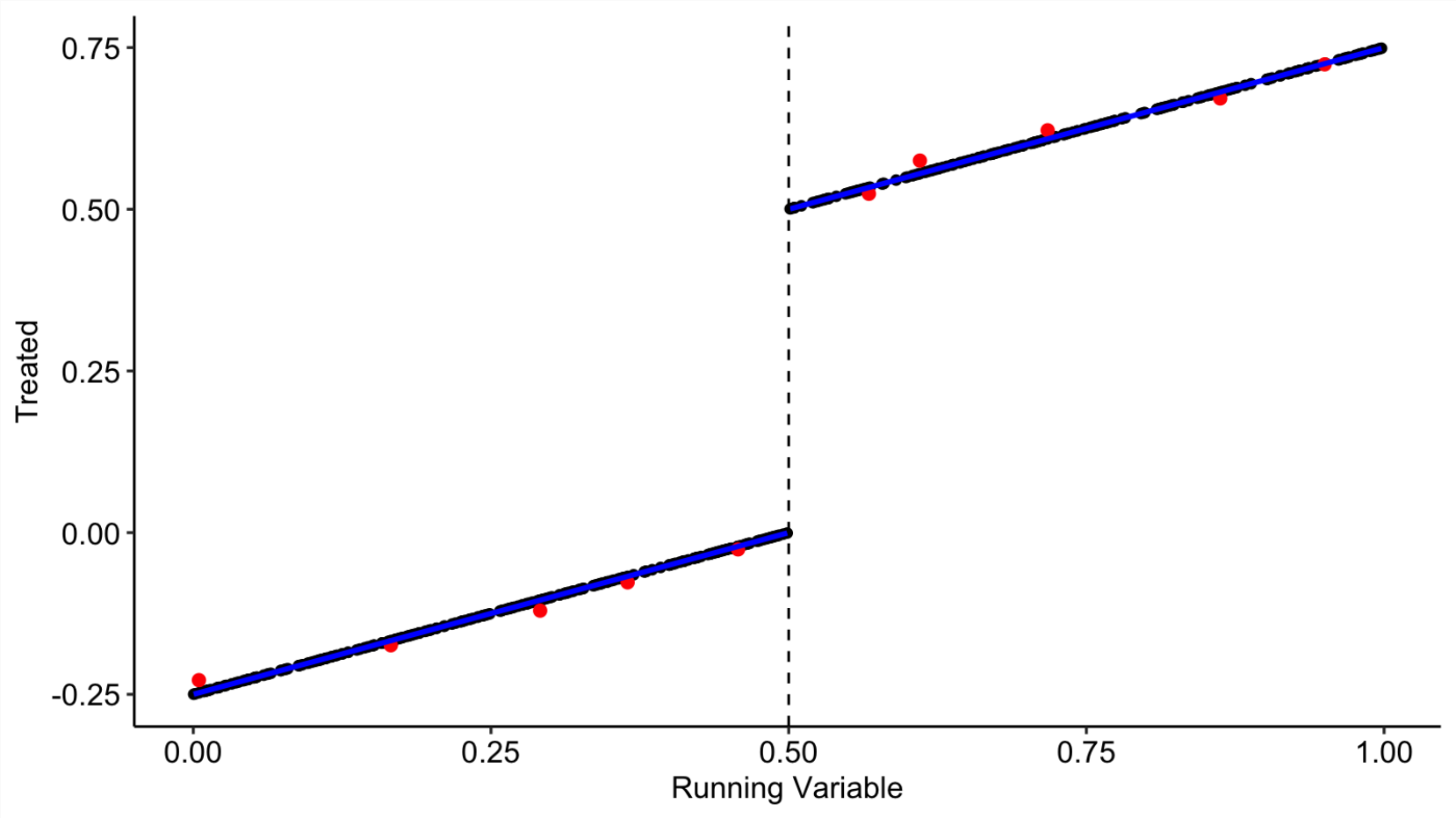
- Let us simulate a dataset with imperfect compliance

```
set.seed(77)
df_fuzzy ← tibble(X = runif(500),
                  above_cut = ifelse(X > 0.5, 1, 0),
                  XC = X - 0.5,
                  treatassign = 0.5*XC + 0.5*above_cut,
                  random = runif(500),
                  Treated = ifelse(treatassign > random, 1, 0),
                  Y = 0.7*Treated + XC + rnorm(500, 0, 0.3))
```

X	above_cut	XC	treatassign	random	Treated	Y
0.4953558	0	-0.0046442	-0.0023221	0.9465440	0	-0.0136400
0.4987442	0	-0.0012558	-0.0006279	0.2720184	0	0.1128971
0.5012444	1	0.0012444	0.5006222	0.6340842	0	0.0068262
0.5035491	1	0.0035491	0.5017745	0.3932667	1	0.6196558

Fuzzy RDD: simulation

- Notice that the y-axis here is not the outcome, it is the percentage treated



Fuzzy RDD: simulation

```
# The true effect is 0.7
without_fuzzy ← lm(Y ~ above_cut*XC, df_fuzzy)
predict_treat ← lm(Treated ~ above_cut*XC, df_fuzzy)
fuzzy_rdd     ← feols(Y ~ 1 | Treated*XC ~ above_cut*XC, df_fuzzy)
```

	Model 1	**Model 2**	**Model 3**
above_cut	0.251***	0.398***	
	(0.069)	(0.060)	
fit_Treated			0.639***
			(0.124)
Num.Obs.	500	500	500
+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001			

RDD: estimation

- The `rdrobust` package has the `rdrobust()` function which runs RDD with
  - optimal bandwidth selection
  - options for fuzzy RD
  - bias correction
  - lots of options (including the addition of covariates)

```
# The true effect is 0.7
library(rdrobust)
m ← rdrobust(df$Y, df$X, c = 0.5)
```

- We can estimate the RDD model by specifying
  - `Y` - a dependent variable
  - `X` - a running variable
  - `c` - a cutoff
  - `p` - the number of polynomials (it applies polynomials more locally than our OLS models do - it avoids weird corner predictions)
  - `h` - a bandwidth size (chosen automatically)
  - `fuzzy` - actual treatment outside of the running variable/cutoff combo (IV)

RDD: estimation

```
## Call: rdrobust
##
## Number of Obs.                  500
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.                  257          243
## Eff. Number of Obs.             123           91
## Order est. (p)                    1            1
## Order bias  (q)                   2            2
## BW est. (h)                   0.212        0.212
## BW bias (b)                   0.304        0.304
## rho (h/b)                     0.697        0.697
## Unique Obs.                     257          243
##
## ================================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ================================================================================
##   Conventional     0.769     0.075    10.288     0.000     [0.623 , 0.916]
##         Robust         -         -     8.875     0.000     [0.600 , 0.941]
## ================================================================================
```

RDD: estimation

- A previous model chose the bandwidth of 0.212
- A common approach to sensitivity analysis is to use
  - the ideal bandwidth
  - twice the ideal
  - half the ideal
  - and see if the estimate changes substantially

```
# The true effect is 0.7
library(rdrobust)
m1 ← rdrobust(df$Y, df$X, c = 0.5, h = 0.212)
m2 ← rdrobust(df$Y, df$X, c = 0.5, h = 2*0.212)
m3 ← rdrobust(df$Y, df$X, c = 0.5, h = 0.5*0.212)
```

```
## [1] 0.7691451 0.7809369 0.7656968
```

RDD: estimation

- Now plot the results
- Note that `rdplot()` uses order-4 polynomial, and `rdrobust()` - local linear regression

```
rdplot(df$Y, df$X, c = 0.5)
```

References

Books

- Huntington-Klein, N. The Effect: An Introduction to Research Design and Causality, Chapter 20: Regression Discontinuity
- Cunningham, S. Causal Inference: The Mixtape, Chapter 6: Regression Discontinuity

Slides

- Huntington-Klein, N. Econometrics Course, Week 08: Regression Discontinuity
- Huntington-Klein, N. Causality Inference Course, Lecture 12: Regression Discontinuity
- Huntington-Klein, N. Causality Inference Course, Lecture 13: Estimating Regression Discontinuity

Articles

- Gelman, A., & Imbens, G. (2019). Why High-Order Polynomials Should not be Used in Regression Discontinuity Designs. Journal of Business & Economic Statistics, 37(3), 447-456.