Untitled

Data and packages

```
library(pacman)
pacman::p_load(tidyverse,
               ggplot2,
               caroline)

# rrp = radical-right parties
rrp <- caroline::read.tab("rrp_rdd.tab") %>%
  dplyr::rename(country_code = iso2c,
                threshold = thrs,
                threshold_l = thrs_l,
                vote = er.v.c,
                vote_l = er.v.c_l,
                parl.presence = er.in,
                parl.presence_l = er.in_l,
                cultural = multic.logit_fd)
```

The story being told here: Assume that parties want to maximize vote share and get to observe citizens' preferences lying in a one-dimensional space. Where should parties position themselves in this space? Downs' median voter theorem implies two factors that can alter the ideological positioning of parties: 1. A change in the distribution of voter preferences 2. A change in other parties' positioning

The first has been studied a lot; the second (the strategic interactions between parties) much less. It implies that parties may change *even though underlying public opinion has not moved at all.*

The idea: can we show that parties react to what other parties are doing, while holding public opinion constant?

The source of exogenous change: institutional rules in many proportional systems that impose a threshold (in % of the popular vote) above which parties gain parliamentary seats. Below the threshold, even though people voted for your party, you don't get any parliamentary seats!

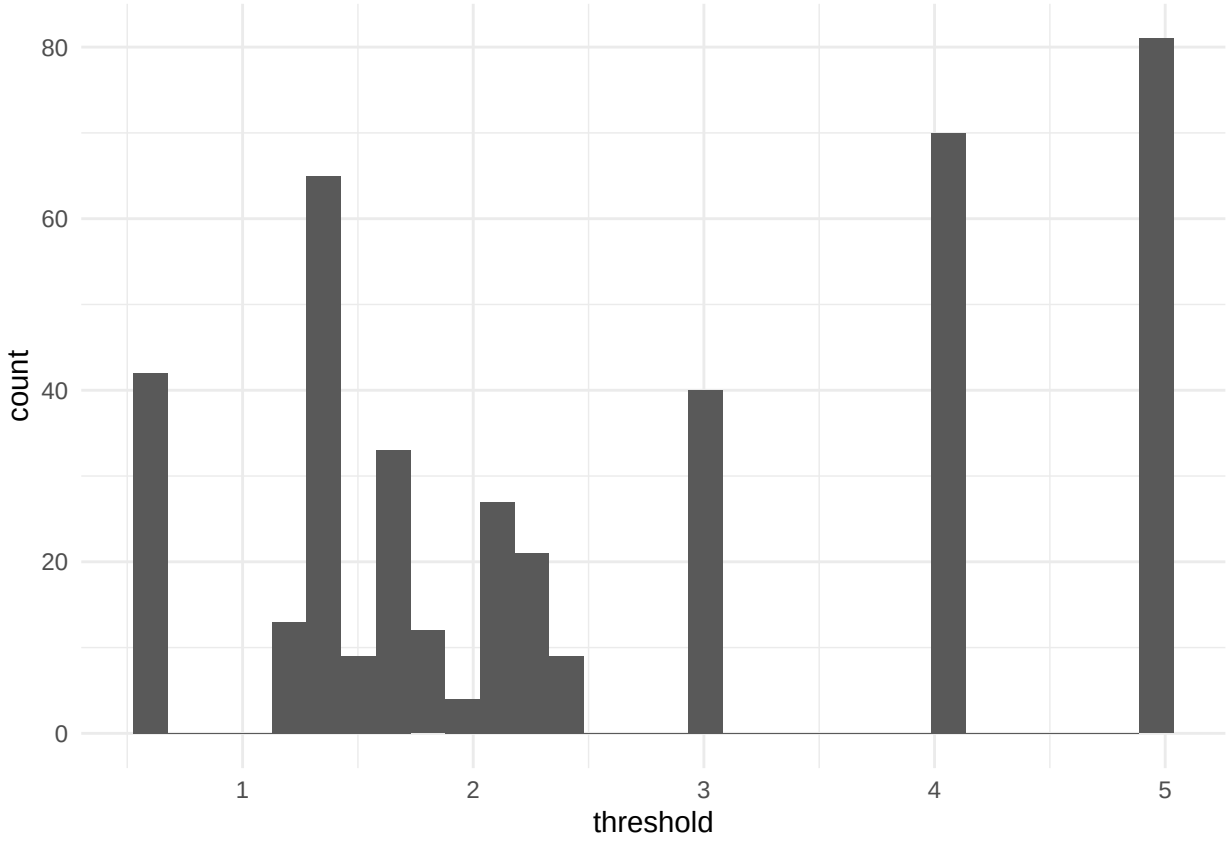
Our data

Our data is at the party-election level. One observation represents one radical-right party in one national legislative election.

What about the threshold? What do they look like?

```
ggplot(rrp, aes(x = threshold)) +
  geom_histogram() +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Hmmm... The rules are not the same everywhere! Is that a problem for identification? Think about our estimand!

We can also see that the rules may change over time in a given country:

```
# What is this code doing? Just looking at the number of unique
# values of "threshold" by counry
rrp %>%
  group_by(country_code) %>%
  summarise(n_distinct(threshold))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 23 x 2
##    country_code `n_distinct(threshold)`
##    <chr>                          <int>
##  1 "\"AT\""                           2
##  2 "\"BG\""                           1
##  3 "\"CH\""                           1
##  4 "\"CZ\""                           1
##  5 "\"DE\""                           1
##  6 "\"DK\""                           2
##  7 "\"EE\""                           1
##  8 "\"ES\""                           1
```

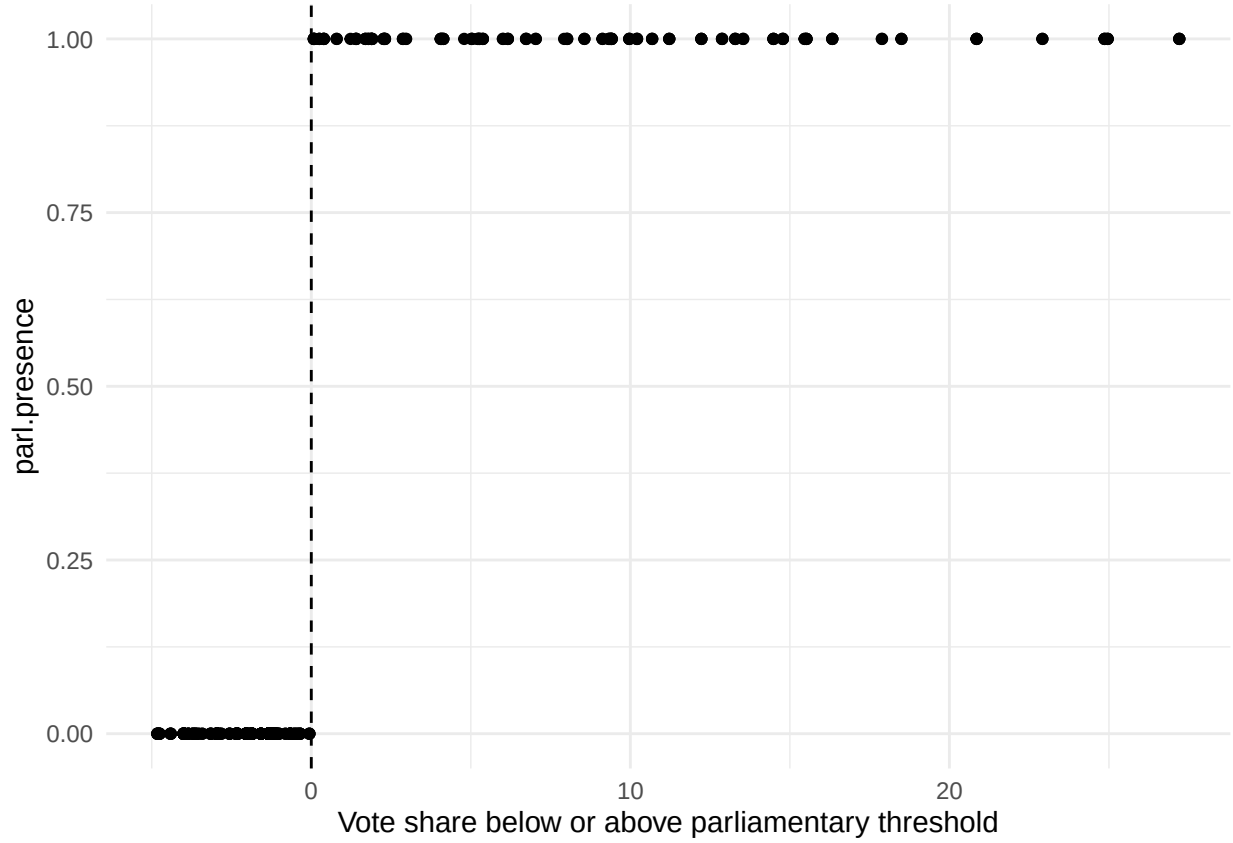
```
##  9 "\"FI\""                          1
## 10 "\"GR\""                          2
## # ... with 13 more rows
```

The rules do change over time in some countries, including in Ireland and Norway where we have three different thresholds over time. Is that a problem?

Treatment assignment mechanism

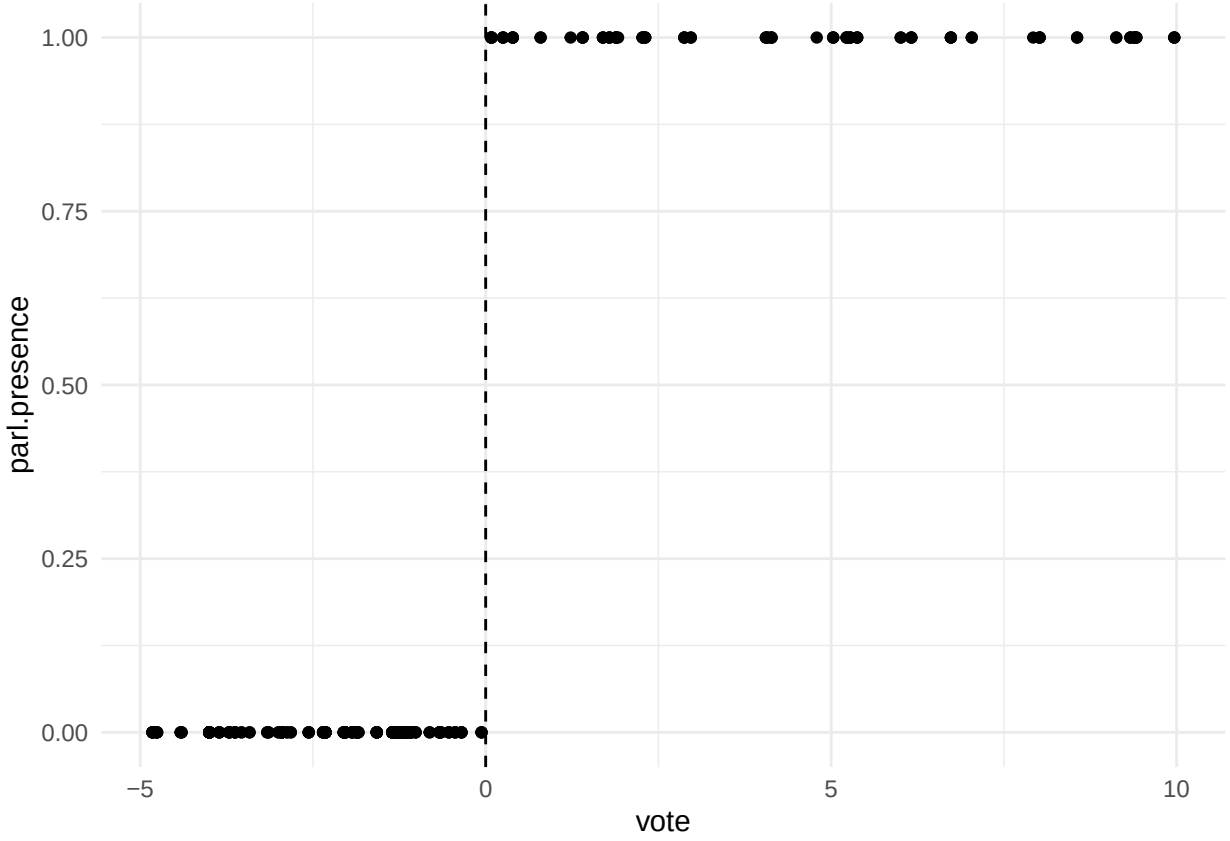
The entire strategy here relies on a discontinuity. So when plotting the running/forcing variable, treatment assignment must visibly change around the threshold. Otherwise, there is no discontinuity; and without a discontinuity, there is no exogeneity to exploit.

```
# With full sample
ggplot(rrp, aes(x = vote, y = parl.presence)) +
  geom_point() +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(x = "Vote share below or above parliamentary threshold") +
  theme_minimal()
```




```
# Let's arbitrarily choose a bandwidth of +/- 10
ggplot(filter(rrp, abs(vote) < 10),
       aes(x = vote, y = parl.presence)) +
  geom_point() +
```

```
  geom_vline(xintercept = 0, linetype = "dashed") +
  theme_minimal()
```



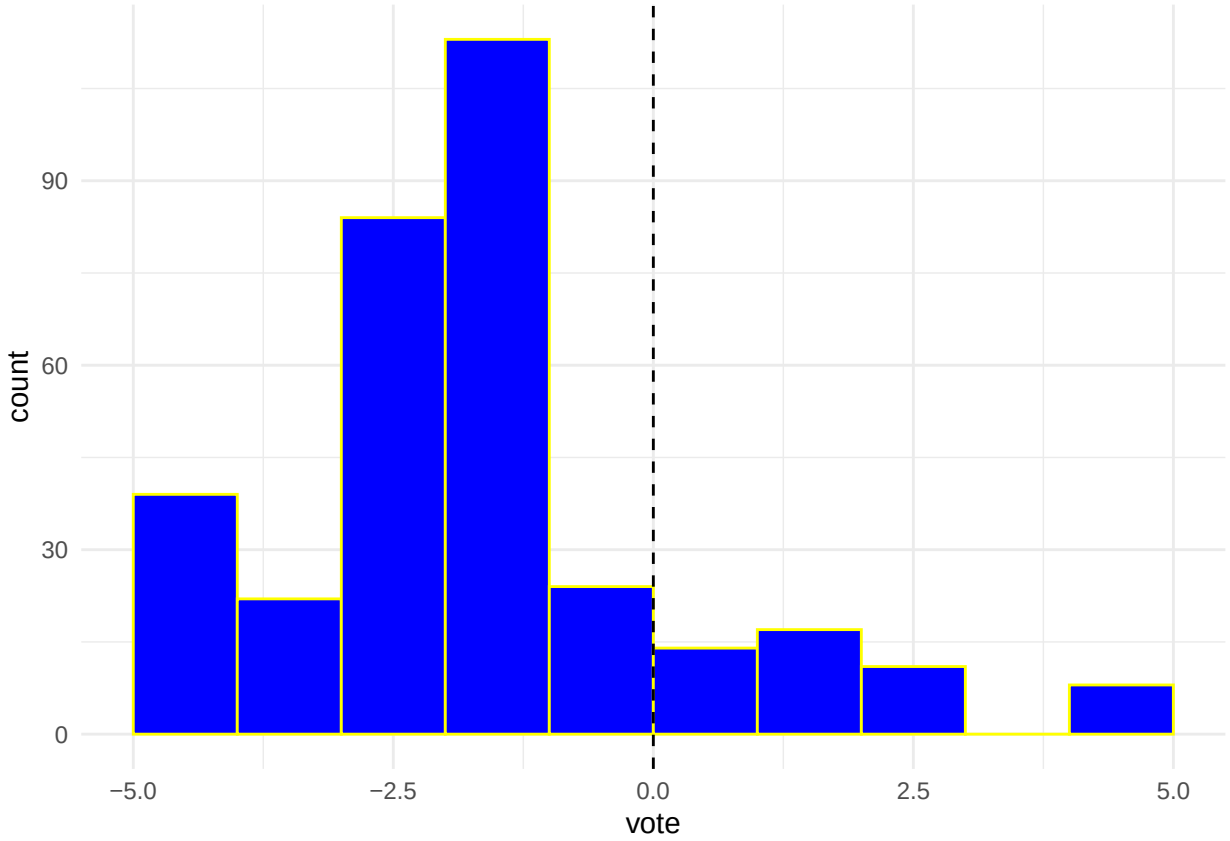
a. What can we tell from this? What sort of RDD is this, and what does that imply?
b. If this had been the other sort of RDD, what sort of complication would arise?

Manipulation at the threshold

Given your substantive knowledge of elections/the European context, what's a potential worry here?

Two strategies to alleviate concerns here: 1. Visual inspection of density of the running variable around the threshold. If there is manipulation, what do we expect to see here? Can there be manipulation on both sides of the threshold? 2. Formal test of the presence of the discontinuity (McCrary)

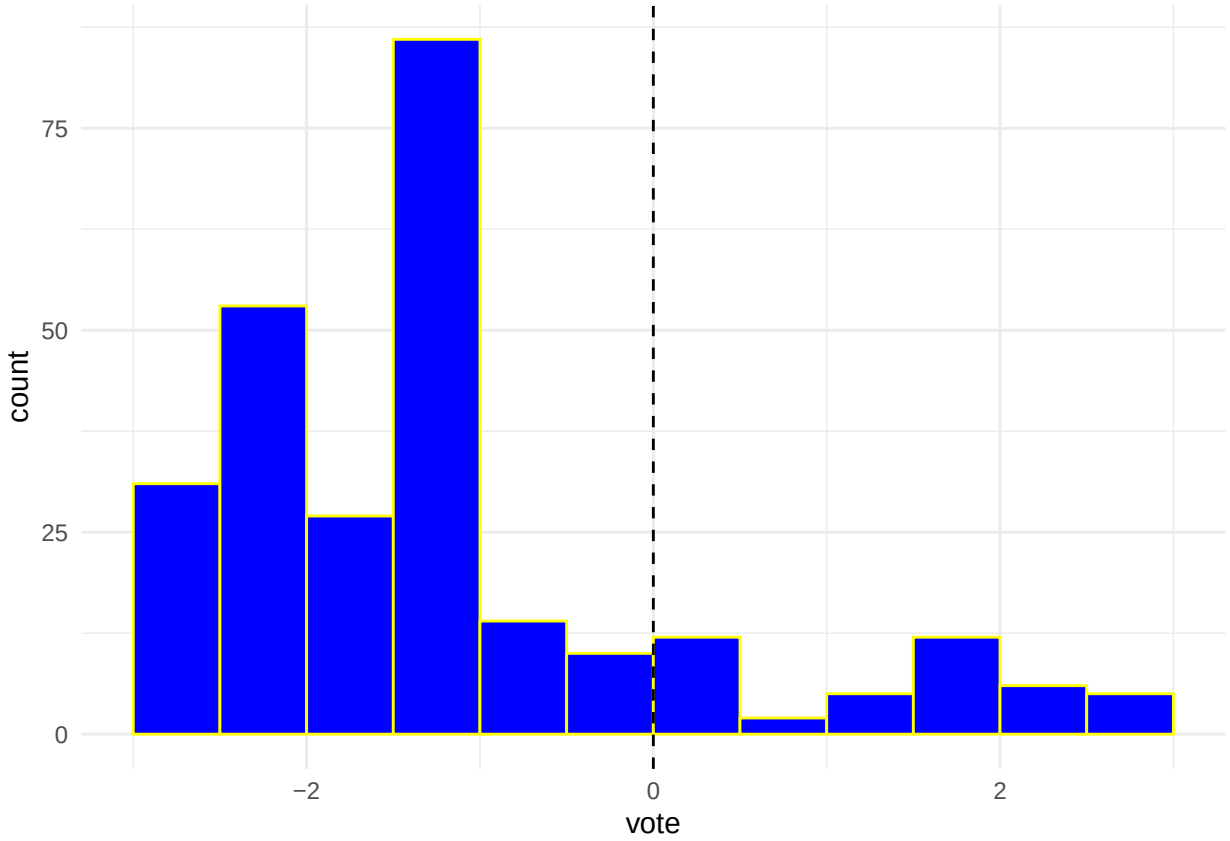
```
# Visual inspection
# be careful here: how we visualize this may change whether or not
# we see any problem. Note "center = 0.5" which avoids bins overlapping
# with integers
ggplot(filter(rrp, abs(vote) < 5), aes(x = vote)) +
  geom_histogram(fill = "blue", col = "yellow",
                 binwidth = 1, center = 0.5) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  theme_minimal()
```



Any thoughts? Two things to think about here: a. the overall distribution is clearly not uniform! is that a problem? why is that? (think substantively about politics here; this is not about stats!) b. what's going on exactly at the threshold?

Let's reduce binwidth and really focus on the threshold:

```
ggplot(filter(rrp, abs(vote) < 3), aes(x = vote)) +
  geom_histogram(fill = "blue", col = "yellow",
                 binwidth = 0.5, center = 0.75) +
    geom_vline(xintercept = 0, linetype = "dashed") +
  theme_minimal()
```



Ok, not clear that we have or don't have a problem... We have some researcher degrees of freedom here!

So let's conduct the formal test developed by McCrary (2008). This is from the "rdd" package.

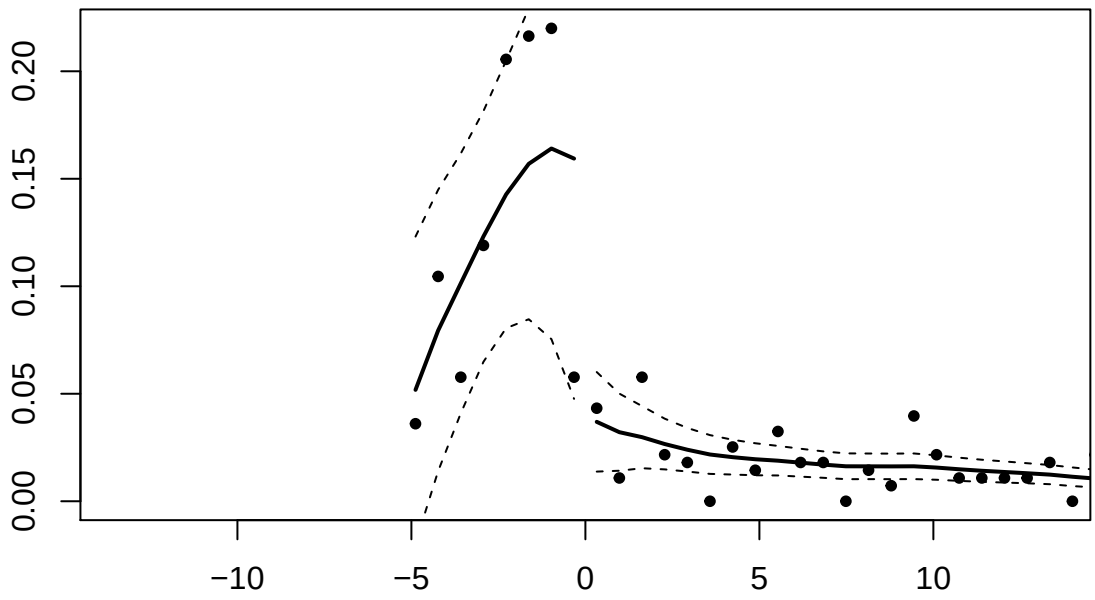
The idea: we have a problem if "the running variable [is] discontinuous at the cutoff, with surprisingly many individuals just barely qualifying for a desirable treatment assignment and surprisingly few failing to qualify." (McCrary 2008: 699)

We'll group points on either side of the threshold into bins (just like in a histogram). Then, we'll run local regressions on each side of the threshold. Our null hypothesis here is that the discontinuity is 0. i.e. there should be roughly as many units just left of the threshold as there are just right of the threshold.

(An aside: sometimes, manipulation of the threshold *is precisely what's interesting*. You can think of this as the opposite of RRD. We call this method "bunching": there is a threshold, often set by administrative rules (e.g. amount of drugs for criminal offense, .08 BAC...), actors know this and change their behavior because of it.)

In the 2008 article, McCracy uses his tests for 1) House elections in the US (failure to reject, which is good and seems to indicate that there's no sorting at the threshold), and 2) roll call votes in the House. So what about this second one? What's the logic?

```
p_load(rdd)
rdd::DCdensity(runvar = rrp$vote, cutpoint = 0)
```




```
## [1] 3.943833e-08
```

Ok, looks like we may have a problem. But this test also uses some arbitrary *binwidth*... And we didn't specify any *bandwidth*...

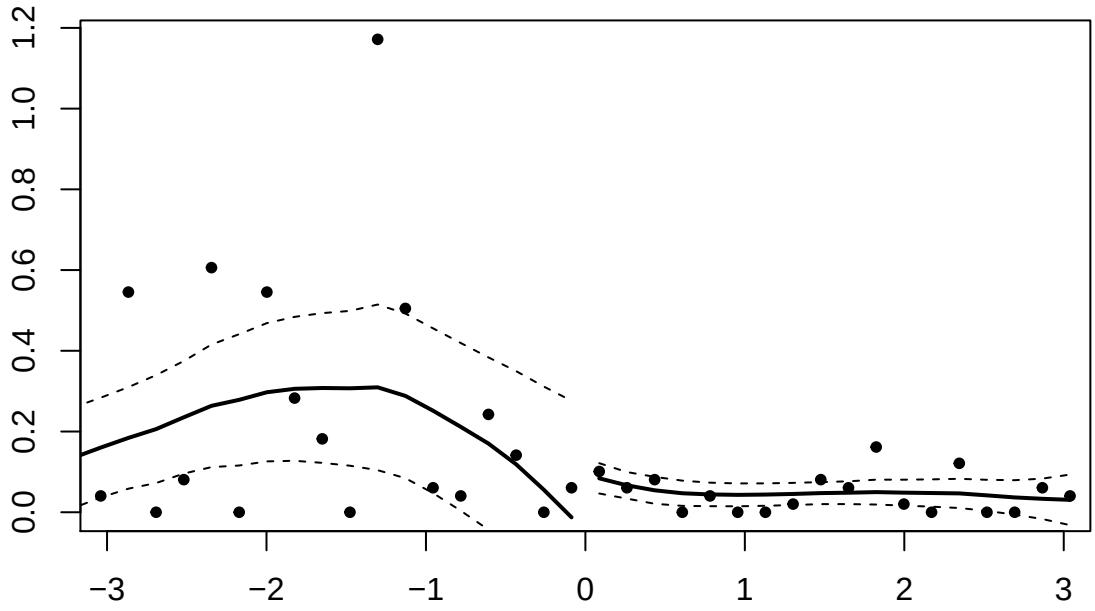
```
runvar <- rrp$vote
# This is the default bindwith used by the McCrary test
2*sd(runvar)*length(runvar)^(-.5)
```

```
## [1] 0.650993
```

```
# Let's simply restrict the bandwidth arbitrarily. We'll use
# party-level observations within 4 points of the threshold
rdd::DCdensity(runvar = filter(rrp, abs(vote)<4)$vote, cutpoint = 0)
```

```
## Warning in log(fhatl): NaNs produced
```

```
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
```




```
## [1] NaN
```

```
# That looks a lot better, as in: there doesn't appear to be
# sorting at the threshold here. (except for that weird bin just
# below -1)

# So what binwidth did the algorithm use here?
# Let's make a simple function that will tell us
runvar_bin4 <- filter(rrp, abs(vote)<4)$vote

mccrary_width <- function(arbitrary_argument_name){
  output <- 2*sd(arbitrary_argument_name)*length(arbitrary_argument_name)^(-0.5)
  paste("The binwidth chosen by the algorithm is", round(output, 3))
}

# Calling our function
mccrary_width(arbitrary_argument_name = runvar_bin4)
```

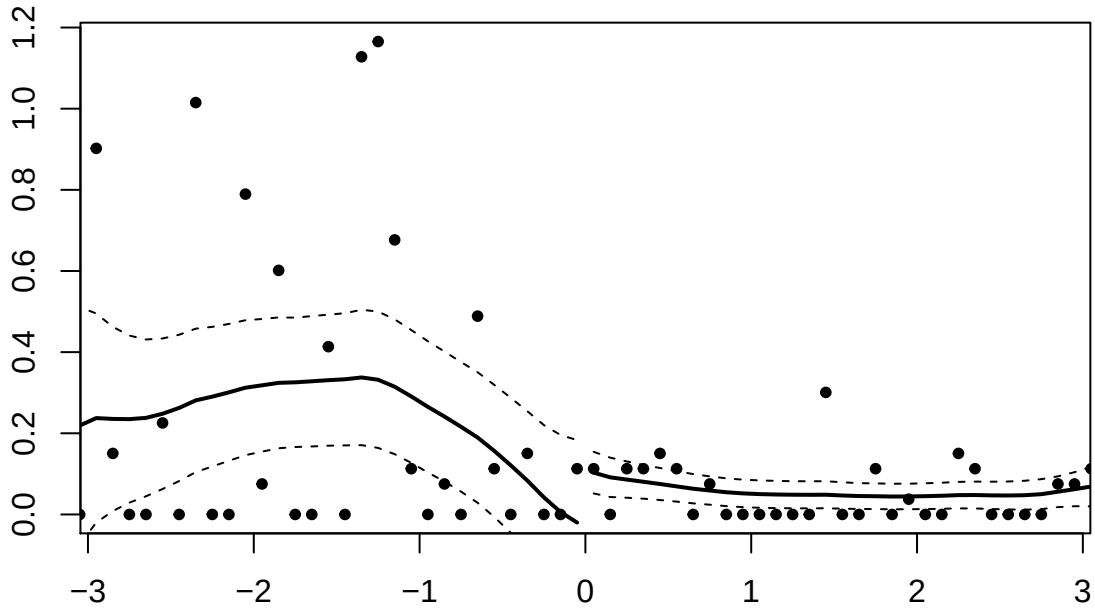
```
## [1] "The binwidth chosen by the algorithm is 0.174"
```

```
#----------------------------------------
# What is the optimal bandwidth?
ik_brandwidth <- IKbandwidth(rrp$vote_l, rrp$cultural, cutpoint = 0,
                             kernel = "triangular")
```

```
rdd::DCdensity(runvar = filter(rrp, abs(vote_l) <= ik_brandwidth)$vote_l,
               cutpoint = 0,
               bin = 0.1)
```

```
## Warning in log(fhatl): NaNs produced
```

```
## Warning in log(fhatl): NaNs produced
```




```
## [1] NaN
```

Graphical presentation of the results

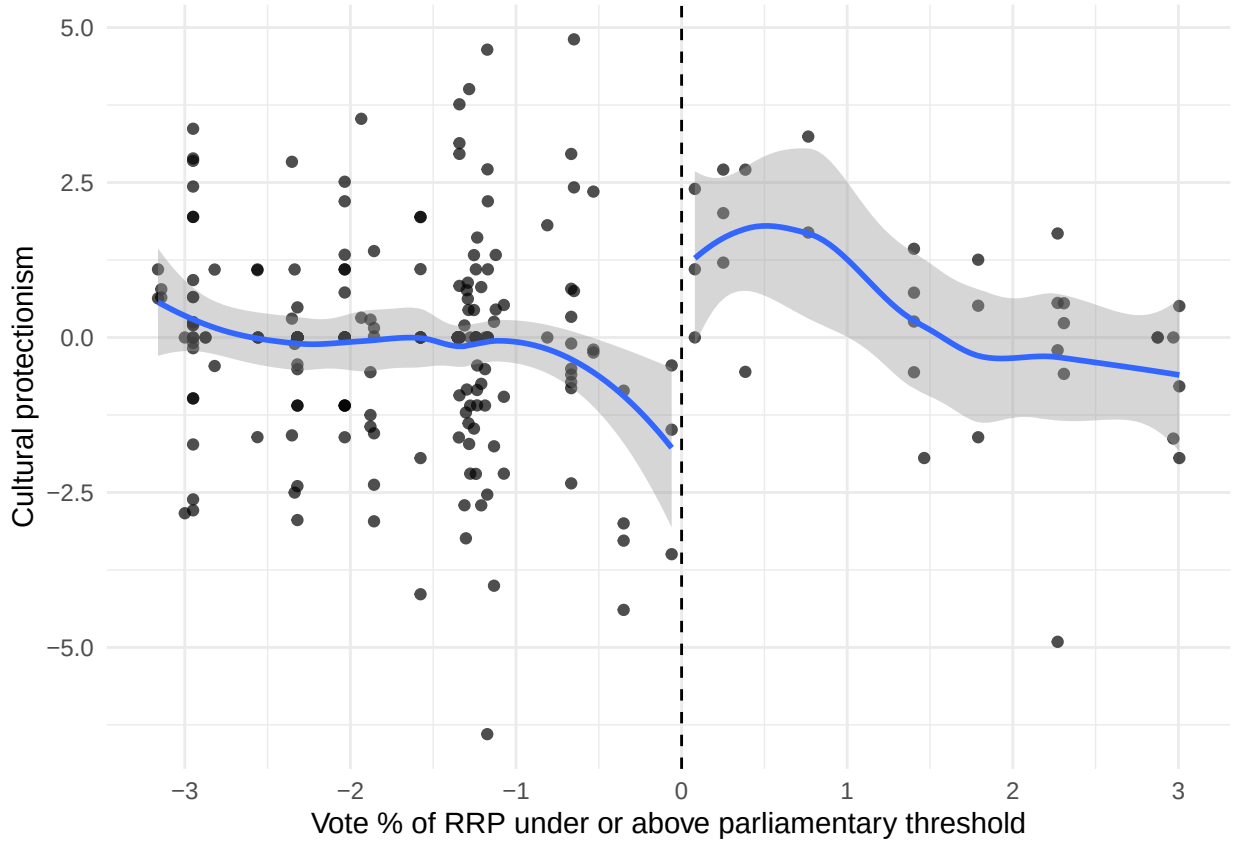
```
# Let's just show the outcome (cultural protectionism) against the running variable
rdd_plot <- ggplot(filter(rrp, vote_l >= -5 & vote_l <= 10),
       aes(x = vote_l, y = cultural, group = parl.presence_l)) +
  geom_point(alpha = 0.7) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(x = "Vote % of RRP under or above parliamentary threshold",
       y = "Cultural protectionism") +
  scale_x_continuous(breaks = seq(-5, 10, 2.5)) +
```

```
  theme_minimal()

# Let's add basic regression lines
# Because I have "group = parl.presence_ll" in code above, I get a piecewise
# regression model out of geom_smooth()
rdd_p1 <-
rdd_plot +
  geom_smooth(method = "lm")

# Looks like we might have a treatment effect...But it's not obvious, is it?
# Let's try with a second-order polynomial
rdd_p2 <- rdd_plot +
  geom_smooth(method = "lm",
              formula = y ~ x + I(x^2))

# 3rd-order?
# This is the main figure in the paper
rdd_p3 <- rdd_plot +
  geom_smooth(method = "lm",
              formula = y ~ x + I(x^2) + I(x^3))

# What about using our "ideal" bandwidth?
ggplot(filter(rrp, abs(vote_l) < ik_brandwidth),
       aes(x = vote_l, y = cultural, group = parl.presence_l)) +
  geom_point(alpha = 0.7) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(x = "Vote % of RRP under or above parliamentary threshold",
       y = "Cultural protectionism") +
  scale_x_continuous(breaks = seq(-4, 3, 1)) +
  geom_smooth() +
  theme_minimal()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 20 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 20 rows containing missing values (geom_point).
```



LATE estimates with regression specifications

```
# Simple  linear model
# Using our optimal bandwith
# note that the authors use country fixed effects
m1_linear <- lm(cultural ~ vote_l + parl.presence_l + country_code,
   data = filter(rrp, abs(vote_l) < ik_brandwidth))

m2_linear_inter <- lm(cultural ~ vote_l*parl.presence_l + country_code,
   data = filter(rrp, abs(vote_l) < ik_brandwidth))

m3_quadratic_inter <- lm(
  cultural ~ vote_l*parl.presence_l + I(vote_l^2) + I(vote_l^2)*parl.presence_l
  + country_code,
  data = filter(rrp, abs(vote_l) < ik_brandwidth)
  )

modelsummary::modelsummary(list(m1_linear, m2_linear_inter, m3_quadratic_inter))
```

	Model 1	Model 2	Model 3
(Intercept)	0.090	0.216	-1.943
	(0.830)	(0.836)	(1.295)
country_code"CZ"	-1.824	-0.579	1.385
	(1.446)	(1.767)	(2.748)
country_code"DE"	-1.100	-0.898	0.304
	(1.099)	(1.110)	(1.233)
country_code"DK"	-0.918	-0.825	-0.997
	(0.883)	(0.885)	(0.883)
country_code"EE"	-1.366	-0.520	0.448
	(1.396)	(1.557)	(1.648)
country_code"ES"	-0.685	-0.680	-0.713
	(0.877)	(0.876)	(0.871)
country_code"FI"	-0.899	-0.887	-0.974
	(0.850)	(0.849)	(0.845)
country_code"GR"	-1.144	-1.083	-0.990
	(0.879)	(0.880)	(0.875)
country_code"IE"	-1.534	-1.408	-1.339
	(0.879)	(0.885)	(0.879)
country_code"LU"	-1.077	-0.950	0.027
	(0.881)	(0.887)	(0.988)
country_code"NL"	-0.770	-0.789	0.059
	(0.889)	(0.888)	(0.965)
country_code"NO"	-1.402	-1.327	-1.556
	(0.926)	(0.927)	(0.927)
country_code"PL"	-1.531	-0.334	1.489
	(1.365)	(1.678)	(2.517)
country_code"PT"	-0.752	-0.743	-0.809
	(0.848)	(0.847)	(0.842)
country_code"RO"	-3.800	-3.438	-3.096
	(1.873)	(1.894)	(1.955)
country_code"SE"	-1.345	-1.275	-1.070
	(1.031)	(1.031)	(1.029)
country_code"SI"	-1.702	-1.293	-0.701
	(1.123)	(1.171)	(1.196)
country_code"SK"	-1.344	-1.550	-1.271
	(1.063)	(1.075)	(1.095)
parl.presence_l"1"	2.501	2.853	4.250
	(0.674)	(0.732)	(1.257)
vote_l	-0.502	-0.395	-3.184
	(0.215)	(0.232)	(1.304)
vote_l:parl.presence_l"1"		-0.681	3.047
		(0.556)	(2.296)
I(vote_l^2)			-0.808
			(0.372)
parl.presence_l"1":I(vote_l^2)			0.362
			(0.881)
Num.Obs.	246	246	246
R2	0.087	0.093	0.113
R2 Adj.	0.010	0.012	0.026
AIC	945.1	945.4	943.9
BIC	1018.7	1022.6	1028.0
Log.Lik.	-451.536	-450.721	-447.941

```r
# Local linear regression with triangular kernel
# We do NOT use the bandwidth here
RDestimate(cultural ~ vote_l, data = rrp)
```

```
##
## Call:
## RDestimate(formula = cultural ~ vote_l, data = rrp)
##
## Coefficients:
##      LATE    Half-BW  Double-BW
##     2.348      2.946      1.530
```