Problem set 6: Mandatory school attendance program

Answer key - PMAP 8521, Spring 2021

March 29, 2021

Contents

There is substantial research and evidence that class attendance has a positive and significant effect on student performance. Because of this, state and local government agencies and school districts have designed programs and policies that incentive students to not miss school days. Examples include tangible prizes like colorful pendants and free tickets to events, automated calls from celebrities, or class policies that mandate attendance.

Existing research has used a range of methods to test the relationship between attendance programs and student performance, including simple regression analysis, randomized experiments, and regression discontinuity approaches.

In this assignment, you will use regression discontinuity approaches to measure the effect of a hypothetical program on hypothetical student grades (this data is 100% fake).

In this simulated program, high school students who have less than 80% attendance during their junior year (11th grade) are assigned to a mandatory school attendance program during their senior year (12th grade). This program requires them to attend school and also provides them with additional support and tutoring to help them attend and remain in school. At the end of their senior year, students take a final test to assess their overall learning in high school.

The dataset I've provided contains four columns:

- `id`: A randomly assigned student ID number
- `attendance`: The proportion of days of school attended during a student's junior year (ranges from 0 to 100)
- `treatment`: Binary variable indicating if a student was assigned to the attendance program during their senior year
- `grade`: A student's final test grade at the end of their senior year

```r
library(tidyverse)
library(rdrobust)
library(rddensity)
library(broom)
library(modelsummary)

# This turns off this message that appears whenever you use summarize():
# `summarise()` ungrouping output (override with `.groups` argument)
options(dplyr.summarise.inform = FALSE)

program <- read_csv("data/attendance_program.csv") %>%
  # Center the running variable at 80
  mutate(attendance_centered = attendance - 80)
```

Step 1: Determine if process of assigning treatment is rule-based

*Was assignment to this program based on an arbitrary rule? Is it a good candidate for a regression discontinuity approach? Why or why not?

From the description of the program, it seems like this is a good candidate for regression discontinuity. A rule clearly determines who gets assigned to the program, which means that people who have attendance right before and right after the threshold are likely comparable, thus creating treatment and control groups.

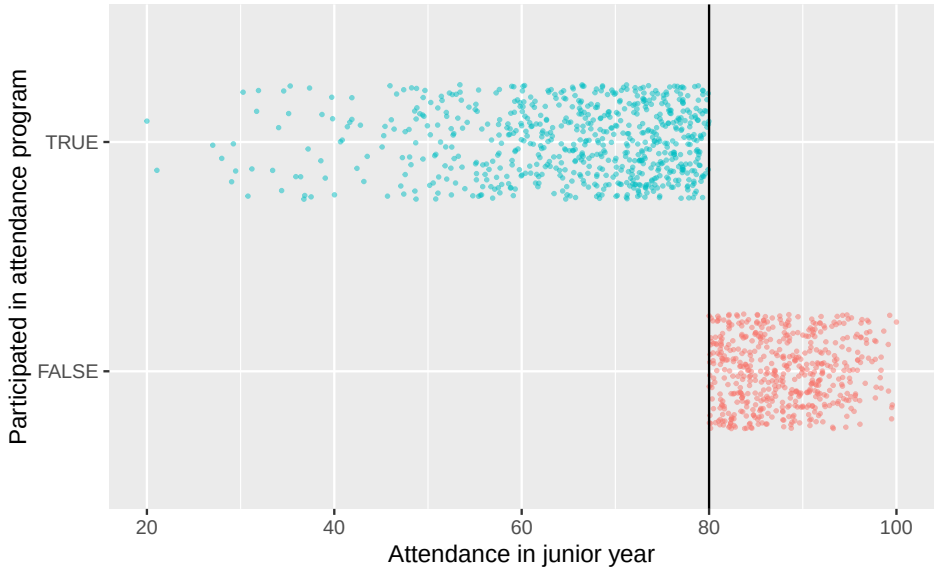
Step 2: Determine if the design is fuzzy or sharp

Make a plot that shows the running variable (`attendance`) on the x-axis and the program indicator variable (`treatment`) on the y-axis. Show the relationship using points (`geom_point`) and color the points by `treatment`.

How strict was the application of the rule? Did any students with attendance above 80% get into the attendance program, or did any students with attendance under 80% not get into the program? Is there a sharp difference in treatment at the cutpoint?

Based on this graph, no students with high attendance participated in the program, and all students with low attendance participated in the program. There's perfect compliance (hooray for nice fake data!), so we can safely say that this is a sharp design.

```
ggplot(program, aes(x = attendance, y = treatment, color = treatment)) +
  geom_point(size = 0.5, alpha = 0.5,
             position = position_jitter(width = 0, height = 0.25, seed = 1234)) +
  geom_vline(xintercept = 80) +
  labs(x = "Attendance in junior year", y = "Participated in attendance program") +
  guides(color = FALSE)  # Turn off the color legend, since it's redundant
```



Step 3: Check for discontinuity in running variable around cutpoint

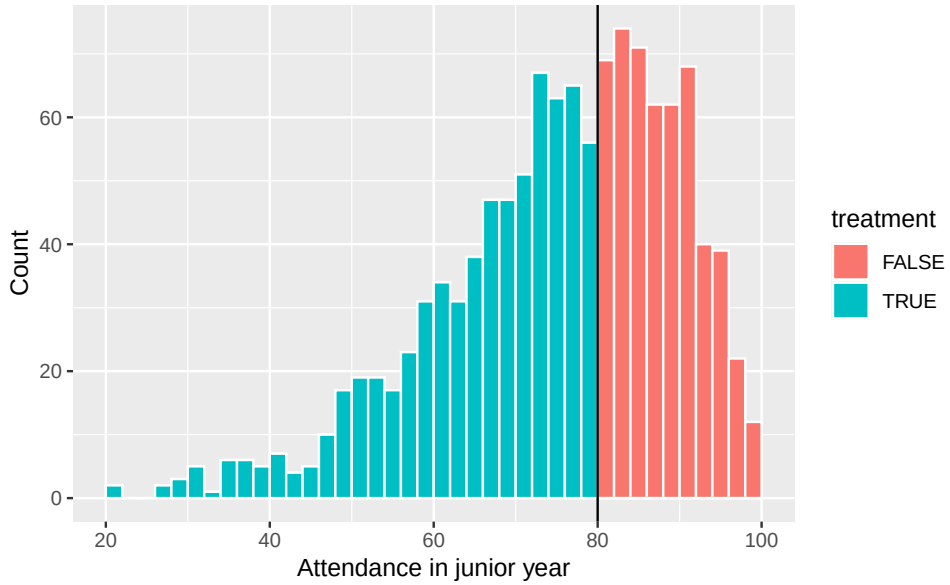
Next, you should check that there was no manipulation in the running variable. We don't want to see a ton of students with 81% or 79% attendance, since that could be a sign that administrators fudged the numbers to either push students into the program or out of the program.

First, make a histogram of the running variable and see if there are any big jumps around the threshold. Fill the histogram by `treatment` and add a vertical line at 80 (`geom_vline(xintercept = 80)`) to show the cutoff. Use an appropriate bin width. If the column near 80 is split into two different colors (it might be, since it might be showing 79 and 80 together), add `boundary = 80` inside `geom_histogram()` to force ggplot to start a bar at 80 and not include 79.

Does it look like there's an unexpected jump in the running variable around the cutoff?

Based on the histogram, there might be an unexpected jump in attendance around 80. There's a dip in attendance right before 80 (in the 78-79 range), which might signal that people are potentially falsely reporting slightly higher attendance to get out of the program. We can test this statistically with a McCrary density test, which shows if the discontinuity in the running variable at the cutoff is significant.

```
ggplot(program, aes(x = attendance, fill = treatment)) +
  geom_histogram(binwidth = 2, color = "white", boundary = 80) +
  geom_vline(xintercept = 80) +
  labs(x = "Attendance in junior year", y = "Count")
```



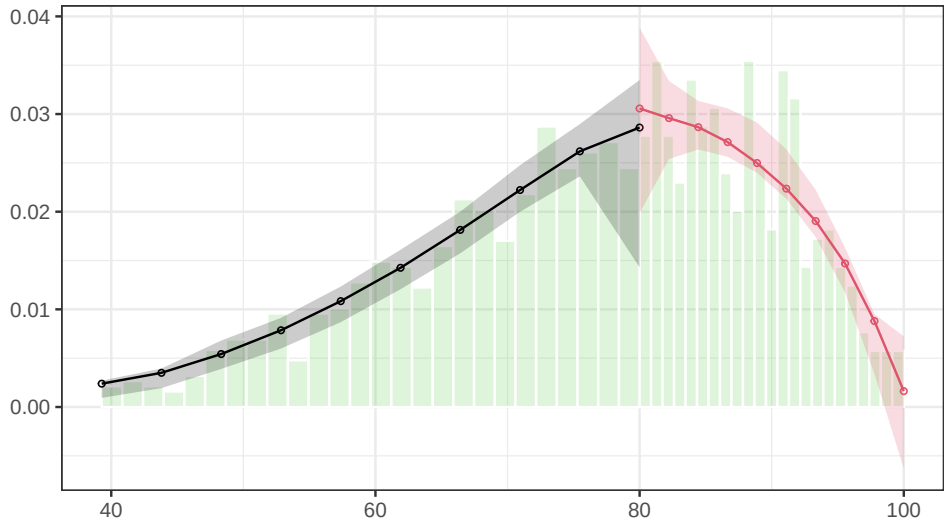
Next, conduct a McCrary density test with `rdplotdensity()` from the `rddensity` library. Refer to the in-class example for the syntax (you'll need to specify `rdd`, `X` (note that it's capitalized), and `type = "both"`). Also, if you don't want two plots to show up when you knit, make sure you assign the output of `rdplotdensity()` to a variable.

Is there a substantial jump at the cutpoint?

According to the figure, there's not really a significant change in the density around the cutpoint—the confidence intervals overlap substantially. If you run `summary(density_jump_stat)` and look at the last line of output, you can see a t-statistic of 0.7748, which has a p-value of 0.4384, which is way higher than 0.05, which means the jump is not statistically significant. We can safely say that the running variable was assigned consistently and there wasn't any manipulation to get people into or out of the program.

```r
density_jump_stat <- rddensity(program$attendance, c = 80)

test_density <- rdplotdensity(rdd = density_jump_stat,
                              X = program$attendance,
                              type = "both")  # This adds both points and lines
```




```r
summary(density_jump_stat)
```

```
##
## Manipulation testing using local polynomial density estimation.
##
## Number of obs =       1200
## Model =               unrestricted
## Kernel =              triangular
## BW method =           estimated
## VCE method =          jackknife
##
## c = 80                Left of c           Right of c
## Number of obs         681                 519
## Eff. Number of obs    384                 421
## Order est. (p)        2                   2
## Order bias (q)        3                   3
## BW est. (h)           13.574              12.521
##
## Method                T                   P > |T|
## Robust                0.7748              0.4384

## Warning in summary.CJMrddensity(density_jump_stat): There are repeated observations. Point estimates
## option massPoints=FALSE to suppress this feature.

##
## P-values of binomial tests (H0: p=0.5).
##
## Window Length / 2          <c     >=c    P>|T|
## 0.290                       6      14    0.1153
## 0.580                      12      23    0.0895
## 0.870                      22      29    0.4011
## 1.160                      31      37    0.5446
```

```
## 1.450                    42      55    0.2229
## 1.740                    52      66    0.2313
## 2.030                    58      70    0.3309
## 2.320                    72      84    0.3785
## 2.610                    74      95    0.1237
## 2.900                    88     103    0.3111
```

Step 4: Check for discontinuity in outcome across running variable

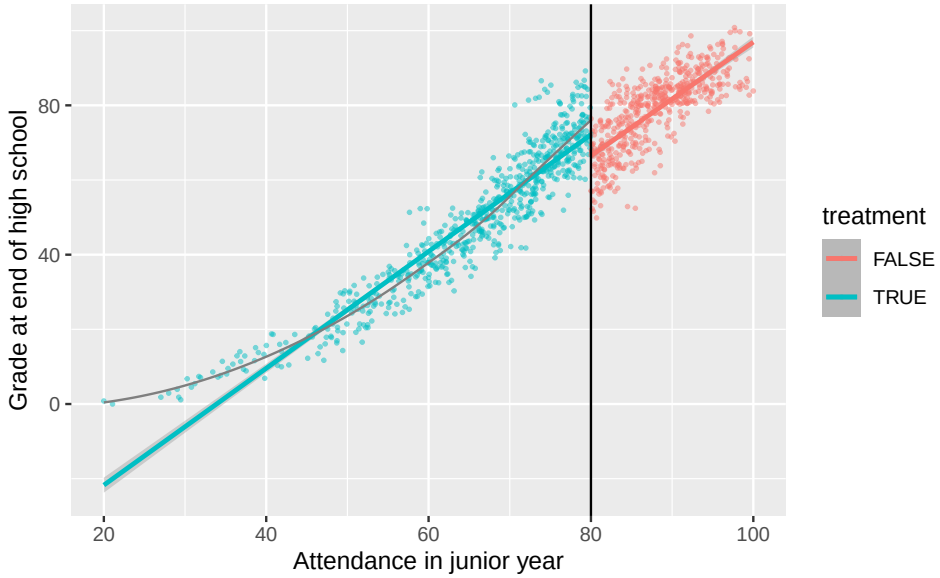
Make a scatterplot with the running variable on the x-axis (`attendance`) and the outcome variable on the y-axis (`grade`), with the points colored by treatment (`treatment`). Make the points small (`size = 0.5` or something similar) and semitransparent (`alpha = 0.5` or something similar) since there are a lot of them. Add a vertical line at the cutoff point. Add two `geom_smooth()` lines: one using data before the cutoff and one using data after the cutoff. Make sure both lines use `method = "lm"`. Refer to the example for the code you need to do this.

Based on this graph, does the program have an effect? Is there a discontinuity in outcome around the cutpoint? Interpret the effect (or non-effect) of the program.

If we look just at this graph, it looks like there might be a jump at the cutoff, but it seems small. People with 79% attendance have much higher average grades than those with 81%. Part of the smallness of this effect is likely due to the shape of the relationship. For those with low grades, the relationship isn't really linear—notice how poorly the straight trendline fits the data where attendance is 20-40%, or where it's 70-80%. Drawing a line across the full data isn't appropriate when we only want to look at a section of the data.

If we include a nonparametric loess trendline, we can see this even better—the curvy trendline fits the data nicely, and the gap at the cutoff is bigger.

```
ggplot(program, aes(x = attendance, y = grade, color = treatment)) +
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth(data = filter(program, attendance < 80), method = "lm") +
  geom_smooth(data = filter(program, attendance < 80), method = "loess",
              size = 0.5, color = "grey50", se = FALSE) +
  geom_smooth(data = filter(program, attendance >= 80), method = "lm") +
  geom_vline(xintercept = 80) +
  labs(x = "Attendance in junior year", y = "Grade at end of high school")
```



Step 5: Measure the size of the effect

Now you need to measure the size and statistical significance of the discontinuity. If there's a jump because of the program, how big is it and how much can we trust it? You'll do this two ways: (1) parametrically with linear regression and (2) nonparametrically with curvy lines and fancy econometrics algorithms built in to the `rdrobust()` function.

Parametric estimation

Create a new dataset based on `program` that has a new variable in it named `attendance_centered`. This will be the value of `attendance` minus 80. This centers student attendance around the cutpoint (if a student had 85% attendance, they'd have a value of 5; if they had 70% attendance, they'd have a value of 10; etc.) and makes it easier to interpret the intercept coefficient in linear models since it shifts the y-intercept up to the cutpoint instead of zero.

```
# I did this back at the beginning of the document after loading the data with
# read_csv(), so I don't need to do it here.
```

Run a regression model explaining `grade` with `attendance_centered + treatment`:

$$\text{Grade} = \beta_0 + \beta_1 \text{Attendance}_{\text{centered}} + \beta_2 \text{Program} + \epsilon$$

Make sure you use the data frame that has your new `attendance_centered` variable.

Interpret the three coefficients. How big is the effect of the program? Is it statistically significant?

```
model_simple <- lm(grade ~ attendance_centered + treatment, data = program)
tidy(model_simple)
```

term	estimate	std.error	statistic	p.value
(Intercept)	66.19	0.330	200.87	0
attendance_centered	1.56	0.020	76.64	0
treatmentTRUE	5.88	0.595	9.89	0

Here's what these coefficients mean:

- β_0**: This is the intercept. Because we centered test scores, it shows the average final score at the 80% threshold. People who had attendance right at threshold scored 66.2 points on average on the final test.**
- β_1**: This is the coefficient for `attendance_centered`. For every percentage point above 80 that people attend school, they score 1.56 points higher on the final test.**
- β_2**: This is the coefficient for the attendance program, and this is the one we care about the most. This is the shift in intercept when `treatment` is true, or the difference between scores at the threshold. Being in the attendance program increases final scores by 5.88 points. This difference is statistically significant ($t = 9.89$; $p < 0.001$).**

Now make two new datasets based on the one you made previously with the `attendance_centered` variable. Filter one so that it only contains observations where `attendance_centered` is between -5 and 5, and filter the other so that it only contains observations where `attendance_centered` is between -10 and 10.

Run the same model (`grade ~ attendance_centered + treatment`) using each of these data frames. Interpret the coefficients. Are they different from the model that uses the complete data?

```
program_bw_10 <- program %>%
  filter(attendance_centered >= -10 & attendance_centered <= 10)
```

```r
program_bw_5 <- program %>%
  filter(attendance_centered >= -5, attendance_centered <= 5)

model_bw_10 <- lm(grade ~ attendance_centered + treatment, data = program_bw_10)
model_bw_5 <- lm(grade ~ attendance_centered + treatment, data = program_bw_5)
```

Put all three models in a side-by-side table with `modelsummary()`. How does the coefficient for `treatment` change across the model specifications? How does the number of observations change? What advantages and disadvantages are there to restricting the data to ±5 or ±10 around the cutpoint? Which program effect do you believe the most? Why?

```r
modelsummary(list("Full data" = model_simple,
                  "Bandwidth = 10" = model_bw_10,
                  "Bandwidth = 5" = model_bw_5))
```

	Full data	Bandwidth = 10	Bandwidth = 5
(Intercept)	66.191	64.195	64.050
	(0.330)	(0.601)	(0.859)
attendance_centered	1.560	2.026	2.148
	(0.020)	(0.097)	(0.272)
treatmentTRUE	5.884	11.869	12.340
	(0.595)	(1.094)	(1.575)
Num.Obs.	1200	640	330
R2	0.907	0.505	0.169
R2 Adj.	0.907	0.503	0.163
AIC	7924.3	4297.8	2228.8
BIC	7944.6	4315.6	2244.0
Log.Lik.	-3958.135	-2144.889	-1110.378
F	5823.048	324.837	33.144

The size of the jump at the cutoff changes as the bandwidth shrinks. If we only look at observations that are ±10 percent around the cutoff, the gap grows a lot to 11.87, which is significant ($p < 0.001$). Narrowing the bandwidth more to ±5 gives us a gap of 12.34 (still significant; $p < 0.001$). We *should* limit the data to only the people right before and after the cutoff—the underlying principle behind regression discontinuity is that the people right before and after are the most comparable. If we run a model with the complete data, we'll be comparing people with 10% attendance and with 99% attendance, and there are clearly underlying differences between those outcomes. Those with 79% and 81%, however, are more comparable and provide us with a better division into treatment and control. So one of the models that uses a bandwidth is better for estimating the size of the effect. Plus, as we saw above, a linear model that uses the full data is inaccurate because the trend isn't linear—the line underestimates the gap at the cutoff.

However, throwing away data outside of the bandwidth has some issues. Notice how the sample size (N) shrinks substantially as the bandwidth shrinks. We move from 1,200 observations in the full model to 640 when the bandwidth is 10, to only 330 when the bandwidth is 5 (!). This makes our standard errors get bigger and can hamper our ability to make valid inferences—note how the standard error grows from 0.595 in the first model to 1.094 when bandwidth = 10 and to 1.575 when bandwidth = 5.

Even so, I'd trust the estimate for the model with the narrowest bandwidth, since it's modeling the program effect for people who are the most similar.

Nonparametric estimation

Next you'll use nonparametric estimation to figure out the size of the gap around the cutpoint. Remember from class that this means we're not using straight lines anymore—we're using curvy lines that don't really have neat $y = mx + b$ equations behind them, so we can't use `lm()` and regular coefficients. Instead, we can use `rdrobust()` to measure the size of the gap.

Use `rdrobust()` with all its default options to find the effect of the program. You'll need to specify `y`, `x`, and `c`. Recall from the in-class example that you'll need to type the name of the variables slightly differently. To refer to the grade column in the program data frame, you'll have to type `program$grade`. Also, make sure you pipe the output of `rdrobust()` to `summary()`, otherwise you won't see the actual program effect (so make sure you type `rdrobust(...) %>% summary()`).

How big of an effect does the program have at the cutpoint? Is the effect statistically significant? Important: if you see a negative number, you can pretend that it's positive. It's negative because the change in trend goes down.

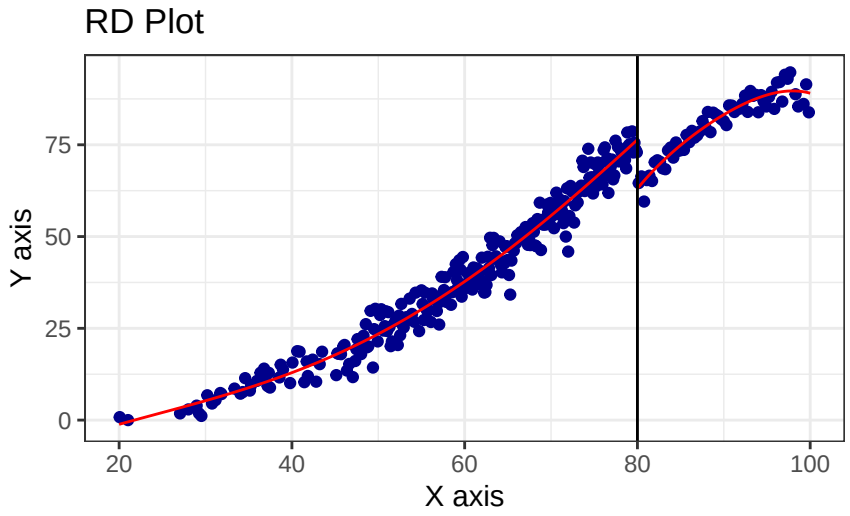
Based on a nonparametric model with a bandwidth of 8.112 and a triangular kernel, there is a 12.013 point gap at the cutoff, and the gap is statistically significant ($z = 8.619$; $p < 0.001$). The plot below shows this effect nicely.

```
rdrobust(y = program$grade, x = program$attendance, c = 80) %>%
  summary()
```

```
## [1] "Mass points detected in the running variable."
## Call: rdrobust
##
## Number of Obs.                 1200
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.                  681         519
## Eff. Number of Obs.             255         279
## Order est. (p)                    1           1
## Order bias  (q)                   2           2
## BW est. (h)                   8.112       8.112
## BW bias (b)                  12.449      12.449
## rho (h/b)                     0.652       0.652
## Unique Obs.                     627         451
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional   -12.013     1.394    -8.619     0.000   [-14.745 , -9.281]
##         Robust         -         -    -7.244     0.000   [-15.473 , -8.883]
## ===============================================================================
```

Make a plot of the effect using `rdplot()`. You'll use the same `y`, `x`, and `c` that you did in `rdrobust()` above.

```
rdplot(y = program$grade, x = program$attendance, c = 80)
```



Nonparametric sensitivity checks

Now that we have an effect, we can adjust some of the default options to see how robust the effect size is.

First we'll play with the bandwidth. Find the ideal bandwidth with with `rdbwselect()`, then run `rdrobust` with twice that bandwidth and half that bandwidth (hint: use `h = SOMETHING`).

```
rdbwselect(y = program$grade, x = program$attendance, c = 80) %>%
  summary()
```

```
## [1] "Mass points detected in the running variable."
## Call: rdbwselect
##
## Number of Obs.                 1200
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.                 681         519
## Order est. (p)                   1           1
## Order bias  (q)                  2           2
## Unique Obs.                    627         451
##
## =============================================================
##                     BW est. (h)    BW bias (b)
##              Left of c Right of c  Left of c Right of c
## =============================================================
##      mserd     8.112      8.112     12.449     12.449
## =============================================================
```

According to `rdbwselect()`, the ideal bandwidth given the data is 8.112, based on the `mserd` criterion, or mean squared error (MSE) approach. See `?rdbwselect` for details about all the possible bandwidth selection procedures.

We can halve that and double that to see how much our estimate changes:

```
# Half bandwidth
rdrobust(y = program$grade, x = program$attendance, c = 80, h = 8.112 / 2) %>%
  summary()
```

```
## [1] "Mass points detected in the running variable."
## Call: rdrobust
##
## Number of Obs.                 1200
## BW type                       Manual
## Kernel                    Triangular
## VCE method                        NN
##
## Number of Obs.                  681         519
## Eff. Number of Obs.             122         146
## Order est. (p)                    1           1
## Order bias  (q)                   2           2
## BW est. (h)                   4.056       4.056
## BW bias (b)                   4.056       4.056
## rho (h/b)                     1.000       1.000
## Unique Obs.                     627         451
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional   -12.761     2.000    -6.380     0.000   [-16.681 , -8.841]
##         Robust         -         -    -3.913     0.000   [-16.492 , -5.485]
## ===============================================================================
```

```
# Double bandwidth
rdrobust(y = program$grade, x = program$attendance, c = 80, h = 8.112 * 2) %>%
  summary()
```

```
## [1] "Mass points detected in the running variable."
## Call: rdrobust
##
## Number of Obs.                 1200
## BW type                       Manual
## Kernel                    Triangular
## VCE method                        NN
##
## Number of Obs.                  681         519
## Eff. Number of Obs.             436         490
## Order est. (p)                    1           1
## Order bias  (q)                   2           2
## BW est. (h)                  16.224      16.224
## BW bias (b)                  16.224      16.224
## rho (h/b)                     1.000       1.000
## Unique Obs.                     627         451
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional   -11.327     0.980   -11.554     0.000   [-13.248 , -9.405]
##         Robust         -         -    -8.613     0.000   [-15.499 , -9.753]
## ===============================================================================
```

Not much changes. The gap is significant in both models. When using the smaller bandwidth (±4.056), the size of the gap is 12.761, and when using the bigger bandwidth (±16.224), it's 11.327.

Next we'll play with the kernel. Use the default ideal bandwidth and adjust the kernel to change how heavily weighted the observations right by the cutoff are. You already used a triangular kernel—that was the first `rdrobust()` model you ran, since triangular is the default. Try using Epanechnikov and uniform kernels (look at the help file for `rdrobust` or look at the in-class example to see how to specify different kernels):

```
# rdrobust() with an Epanechnikov kernel
rdrobust(y = program$grade, x = program$attendance,
         c = 80, kernel = "epanechnikov") %>%
  summary()
```

```
## [1] "Mass points detected in the running variable."
## Call: rdrobust
##
## Number of Obs.                 1200
## BW type                       mserd
## Kernel                   Epanechnikov
## VCE method                       NN
##
## Number of Obs.                  681         519
## Eff. Number of Obs.             245         261
## Order est. (p)                    1           1
## Order bias  (q)                   2           2
## BW est. (h)                   7.780       7.780
## BW bias (b)                  12.498      12.498
## rho (h/b)                     0.622       0.622
## Unique Obs.                     627         451
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional   -11.910     1.377    -8.649     0.000   [-14.609 , -9.211]
##         Robust         -         -    -7.313     0.000   [-15.348 , -8.860]
## ===============================================================================
```

```
# rdrobust() with a uniform kernel
rdrobust(y = program$grade, x = program$attendance,
         c = 80, kernel = "uniform") %>%
  summary()
```

```
## [1] "Mass points detected in the running variable."
## Call: rdrobust
##
## Number of Obs.                 1200
## BW type                       mserd
## Kernel                      Uniform
## VCE method                       NN
##
## Number of Obs.                  681         519
## Eff. Number of Obs.             195         231
## Order est. (p)                    1           1
## Order bias  (q)                   2           2
## BW est. (h)                   6.441       6.441
## BW bias (b)                  11.081      11.081
## rho (h/b)                     0.581       0.581
## Unique Obs.                     627         451
```

```
##
## ===============================================================================
##         Method     Coef. Std. Err.          z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional   -11.531     1.448    -7.965     0.000   [-14.368 , -8.694]
##         Robust         -         -    -6.817     0.000   [-15.171 , -8.395]
## ===============================================================================
```

Changing the kernel does very little to influence the size of the gap. With an Epanechnikov kernel (and an ideal bandwidth of 7.780), the gap is 11.910 points ($p < 0.001$), and with a uniform kernel (and an ideal bandwidth of 6.441), it is 11.531 points ($p < 0.001$). The measured program effect is thus not dependent on the kernel specification.

Step 6: Compare all the effects

Make a list of all the effects you found. Which one do you trust the most? Why?

Write them in this table if it's helpful:

Method	Bandwidth	Kernel	Estimate
Parametric	Full data	Unweighted	5.884
Parametric	10	Unweighted	11.869
Parametric	5	Unweighted	12.340
Nonparametric	8.112	Triangular	12.013
Nonparametric	4.056 (half)	Triangular	12.761
Nonparametric	16.224 (double)	Triangular	11.327
Nonparametric	7.780	Epanechnikov	11.910
Nonparametric	6.441	Uniform	11.531

Does the program have an effect? Should it be rolled out to all schools? Why or why not?

The program has a definite effect, causing an increase of 12ish points in the final grade for those around the cutoff (the local average treatment effect, or LATE). The estimate is lowest when using a parametric model on the full data, since the relationship between attendance and grades is not linear, which makes it so the linear trend underestimates the size of the gap. The estimate more consistent when using either parametric or nonparametric models with a bandwidth, since those fit the data around the cutoff specifically. The nonparametric estimate changes very little across different bandwidths and kernels.

Whether it should be rolled out to all schools is a different question. There is clear causal evidence that the program has a substantial and significant LATE. However, actually rolling out the program depends on more than just evidence—there are politics involved, questions of funding, etc. Also, because this is the LATE, it's only really valid for students in the bandwidth area (i.e. with attendance of 80% ± 8ish). If a school has higher or lower average attendance, the causal evidence we have doesn't apply to it.