# Fuzzy Regression Discontinuity

### Data

```
#install.packages("downloadthis")
library(downloadthis)

download_link(
  link = "https://bayreuth-politics.github.io/CI22/data/enter.csv",
  output_name = "enter",
  output_extension = ".csv",
  button_label = "Lab 9 Data",
  button_type = "success",
  has_icon = TRUE,
  self_contained = TRUE
)

#download_link(
#  link = "https://github.com/dpir-ci/CI22/raw/gh-pages/docs/lectures/lecture8.pdf",
#  output_name = "week8",
#  output_extension = ".pdf",
#  button_label = "Lecture Slides",
#  button_type = "default",
#  has_icon = FALSE,
#  self_contained = FALSE
#)
```

---

# Recap

## Sharp RDD

Last week we discussed sharp regression discontinuity designs, in which a cut-off perfectly determines the treatment status of units. In simple words: all units on one side of the cut-off are treated, while all units on the other side are not:

$$\Pr\left(D_i = 1\right) = p\left(X_i\right)$$

$$p\left(X_i\right) = \begin{cases} 1 & \text{if } X_i \geq X_0 \\ 0 & \text{if } X_i < X_0 \end{cases}$$

## Fuzzy RDD

While sharp RDDs are only a special case of regression discontinuity, RDDs are more powerful than that. **Fuzzy RDDs** allow us to estimate causal effects in settings where treatment assignment is not deterministic. The cut-off may not perfectly determine treatment exposure, but it creates a *discontinuity in the probability* of treatment exposure. The cut-off therefore does not perfectly predict treatment status. Still, we can use a fuzzy RDD as long as it introduces a meaningful as-if random discontinuity that we can take advantage of.

$$\lim_{X_i \uparrow X_0} p\left(X_i\right) \neq \lim_{X_i \downarrow X_0} p\left(X_i\right)$$

As we learned before, we cannot simply ignore treatment uptake as it likely is not random: Other (unobserved) covariates might come into play and cause units to take up or reject treatment. In other words, *non-compliance* with treatment assignment is an issue in this case that must be taken into account in order to estimate an unbiased causal effect - this should have you recall Lab 7.

In fact, a *fuzzy RDD* simply incorporates an instrumental variable (IV) that is applied to a regression discontinuity. Usually we can use the treatment assignment - i.e. on which side a unit is of a cut-off - as an instrument for treatment status. Similarly to the IV design, we estimate the *LATE* among compliers - or *CACE* at the discontinuity. Note that all assumptions of the

IV design need to be satisfied here, too. Extrapolating across the discontinuity, this allows us to estimate the *LATE* for compliers at the discontinuity (close to the threshold):

$$\begin{aligned}
\alpha_{FRDD} &= E\left[Y_1 - Y_0 \mid X = c\right] \\
&= \frac{\lim_{x \downarrow c} E[Y \mid X = c] - \lim_{x \uparrow c} E[Y \mid X = c]}{\lim_{x \downarrow c} E[D \mid X = c] - \lim_{x \uparrow c} E[D \mid X = c]} \\
&= \frac{\text{outcome discontinuity}}{\text{treatment discontinuity}} \\
&\approx \frac{E[Y \mid Z = 1] - E[Y \mid Z = 0]}{E[D \mid Z = 1] - E[D \mid Z = 0]}
\end{aligned}$$

This again should sound familiar - the *LATE* among compliers in fuzzy RDDs can be estimated using a 2SLS regression as we will see in today's Lab.

---

**Before starting this seminar**

1. Create a folder called "lab9"
2. Download the data (you can use the button or the one at the top, or read csv files directly from github):
3. Open an R script (or Markdown file) and save it in our "lab9" folder.
4. Set your working directory using the setwd() function or by clicking on "More". For example *setwd("~/Desktop/Causal Inference/2022/Lab9")*
5. Let's install an load packages that we will be using in this lab:

```
library(rdd) # to conduct a density test
library(rdrobust) # to conduct non-parametric estimation
library(rddensity) # to conduct a density test
library(tidyverse)  # ggplot(), %>%, mutate()
library(broom)  # Convert models to data frames
library(rdrobust)  # For robust nonparametric regression discontinuity
library(estimatr)  # Run 2SLS models in one step with iv_robust()
library(modelsummary)  # Create side-by-side regression tables
library(kableExtra)  # Fancy table formatting
library(ggplot2) # to generate plots
library(haven) # to import dta files.
library(wesanderson) # To change the colour scheme of ggplot
```

# Seminar Overview

In this **seminar**, we will cover the following topics:
1. Familiarise ourselves with the data structure of fuzzy RDD and the substantive interpretation of the estimator.
2. Visualise discontinuities in terms of treatment and outcome at the cut-off.
3. Estimate parametric fuzzy RDDs using 2SLS regressions.
4. Using the `rdrobust` package to estimate non-parametric fuzzy RDDs.
5. Conduct the following robustness/falsification tests such as balance test, placebo outcome and density test.

---

# Staying in the First League: Parliamentary Representation and the Electoral Success of Small Parties

To familiarize ourselves with fuzzy RDDs, we will be analysing data from the paper Staying in the First League: Parliamentary Representation and the Electoral Success of Small Parties authored by Elias Dinas, Pedro Riera and Nasos Roussias.

In this paper, the authors seek to contribute to answering one of the most essential research questions in party politics: Why are some small parties successful whereas other whither away? What accounts for the variation in the trajectories of small parties?

The authors theorise an organisational mechanism that contributes to parties' success. According to their expectations, *entering parliament* should lead to an increase in a party's vote share in the subsequent election. This is because being represented in parliament usually comes with a lot of benefits, which are not experienced by parties that are not represented: influence on policy-making, public funding, media visibility, an increase in organisational capacity (staff, etc.) as well as reduced uncertainty about electoral viability and ideological profiles. Conversely, parties that fail to enter parliament do not gain these benefits and might even be abandoned by promising personnel. As you notice, there might be a lot of reasons which make it difficult to isolate a causal effect without an appropriate identification strategy.

In many multi-party systems, there is a legal threshold for representation in parliament at the national level, usually in terms of a fixed vote share, that determined whether a party enters parliament or not. The authors exploit the randomness introduced by these electoral thresholds to circumvent endogeneity problems. Clearly, parties are not able to manipulate their vote share - at least in democracies. The authors look at countries that had employed a legal threshold of representation at the national level at least one since 1945.

**Why do we need to estimate a *fuzzy RDD* here?** Unlike the sharp RDD, in this case some candidates make it to parliament even though their party did not pass the threshold - in other words, there is non-compliance with treatment assignment as the discontinuity does not fully determine treatment status. Roughly 20% of electoral systems allow some parties to get into parliament even if they did not meet the threshold - can you think of reasons why?

The authors use a linear regression estimator with a triangular kernel and optimal bandwidths determined by Imbens/Kalyanaramn's algorithm (h=2.4, after the cut-off is centred). The outcome is parties' vote share at the subsequent election ($t_1$). They find a local average

treatment effect among compliers of 1.9%-points: Parties that have not cleared the threshold received 3.9 percent at election $t_1$, whereas those who overcome the threshold get 5.8 percent. This amounts to a meaningful and substantive effect of an increase of around 40% in their vote share.

*Note: Electoral thresholds vary by country, ranging from 0.67%-points to 10%-points. To make observations comparable, the authors standardise electoral results. They use 3.51 as common cut-off point and transform vote shares to that scale. Importantly, the relative distance from the threshold remains the same:*

$$\frac{VoteShare_i}{Threshold_i} = \frac{VoteShare_{i,standardised}}{3.51}$$

We will be using the following variables:

| Variable | Description |
|---|---|
| `performance` | This is the *standardised and centred* vote share of party $i$ at $t_0$. |
| `treat` | Binary variable that indicates whether the party's vote share met/exceeded the threshold (=1) or not (=0). |
| `newsuccess` | The party's vote share in the subsequent election at $t_1$. |
| `treated` | Binary variable indicating whether a party entered parliament (=1) or not (=0) |
| `oldsuccess` | The *standardised and centred* vote share of party $i$ at $t_{(-1)}$, which we'll use as a placebo. |
| `newperc` | Indicates the standardised vote share in absolute terms |
| `seats` | Indicates the number of seats won in an election |
| `established` | Binary variable indicating if a country is an established democracy |
| `oldparty` | Count variable indicating how many elections a party had contested before. |

---

Now let's load the data. There are two ways to do this:

You can load the brands dataset from your laptop using the `read.csv()` function. You can call this data set `enter`.

```
# Set your working directory
#setwd("~/Desktop/Causal Inference/2022/Lab9")
#
library(readr)
enter <- read.csv("https://bayreuth-politics.github.io/CI22/data/enter.csv")
```

```
#head(enter)
```

---

Now, let's see how the data looks like. Generate a plot using the `rdplot()` function from the `rdrobust` package.

The syntax is quite simple - see it:

**Exercise 1: Generate a plot using the `rdplot(Y, X, ci = 95, binselect = "", subset = condition)` function. Replace X with `enter$performance` and Y with `enter$newsuccess`. Set the argument `ci` equal to 95, so we can display confidence interval in each bin. Let's create a RD plot with evenly-spaced bins by setting the argument `binselect` equal to "es". Finally, let's subset the data for parties whose standardised vote share was below 3.51. For this, set the argument `subset` equal to `performance < 3.51`. For more details about this function please see below.**

```
rdplot(Y, X, c = 0, ci = 95, binselect = "es", subset = Z < condition)
```

| Function/argument | Description |
|---|---|
| Y | Y is the dependent variable. |
| X | X is the running variable |
| c | To specify the cut-off the default is 0 |
| ci | Optional graphical option to display confidence intervals of selected level for each bin |
| binselect | Specifies the procedure to select the number of bins. See options below:<br>**es: IMSE-optimal evenly-spaced method using spacings estimators.**<br>esmv: mimicking variance evenly-spaced method using spacings estimators. This is the default option.<br>qspr: IMSE-optimal quantile-spaced method using polynomial regression. |
| subset | an optional vector specifying a subset of observations to be used. |

*Reveal Answer*

```
rdplot(enter$newsuccess, enter$performance, ci=95, binselect="es", subset = enter$performa
```

Note that it might make sense to look at the discontinuity in terms of the treatment, too. This will give us a (visual) indication of whether the discontinuity in terms of treatment status is meaningful. You can simply use the `rdplot` command to do this.

```
rdplot(enter$treated, enter$performance, ci=95, binselect="es", subset = enter$performance
```

As expected, we have perfect compliance above the threshold - and rather strong yet imperfect compliance below the threshold.

Note that we use 3.51 to drop observations larger than that number (weighted threshold). This insures a balanced N below and above the cut-off. While the running variable is standardised, it cannot take on values smaller than 0 (which would correspond to a 0% vote share). To account for this, we mirror this natural boundary above the threshold.

Let's check that this indeed a fuzzy design. To do so let's first create a new variable called `treatment_received`.

**Exercise 2: Create a new variable that is equal to "Treated" if the `treated` variable is equal to 1, and "Not treated" otherwise. You can use the `mutate()` and `ifelse()` function to do this. You can see an example below.**

```
data <- data %>%
  mutate(new_variable = ifelse(variable == 1, "Treated", "Not treated"))
```

*Reveal Answer*

```
enter <- enter %>%
  dplyr::mutate(treatment_received = ifelse(treated == 1, "Treated", "Not treated"))

table(enter$treatment_received)
```

Now that we have created this variable, let's use it to generate other RD plots that will help us identify whether the units actually received the treatment condition that they were assigned to. Let's use ggplot this time.

**Exercise 3: Generate the following plot:**

- Map data components into the graph `ggplot(aes(x = running variable, y = outcome, colour = as.factor(new received variable)), subset(subset=running variable <3.51), data = data)`
- Add a scatter plot by adding the following function: `geom_point(size = , alpha = )`. Set the `size` argument equal to 1 and the `alpha` argument equal to 0.6.
- Add a linear regression below the threshold using: `geom_smooth(data = filter(subset(data,subset=ru variable <3.51), running variable <= 0), method = "lm")`
- Add a linear regression above the threshold using: `geom_smooth(data = filter(subset(data,subset=ru variable <3.51), running variable > 0), method = "lm")`
- Add vertical that pass through 0 on the vertical axis by setting the `intercept` equal to 0: `geom_vline(xintercept = 0)`
- Change the labels on the y and x axis using: `labs(x = "Vote share t0", y = "Vote share t1", color = "Treated")`
- Change the default colours adding the Wes Anderson palette: `scale_color_manual(values=wes_palette`

*Reveal Answer*

```
ggplot(subset(enter, subset=performance<3.51), aes(x = performance, y = newsuccess, color
  # Make points small and semi-transparent since there are lots of them
  geom_point(size = 1, alpha = 0.5) +
  #  Add a line based on a linear model for the parties meeting the threshold
  geom_smooth(data = filter(subset(enter,subset=performance <3.51), performance <= 0), met
  # Add a line based on a linear model for the parties missing the threshold
 geom_smooth(data = filter(subset(enter,subset=performance <3.51), performance > 0), metho
  # Add labels
  geom_vline(xintercept = 0) +
  labs(x = "Vote share t0", y = "Vote share t1", color = "Treated") +          scale_color
```

From a quick visual inspection, we can see that we are dealing with a case of one-sided non-compliance. This means that some parties that were below the threshold entered parliament

nonetheless - this is what we would expect. Conversely, every party that met its respective threshold entered parliament - as it should be.

Let's now calculate the proportion of compliers above and below the threshold, just to know more about the extent of non-compliance we are dealing with.

**Exercise 4: Calculate the proportion of compliers above and below the threshold. Use the instructions and functions below to do this.**

```
data %>%
  group_by(variable1, variable2) %>%  #
  summarise(count = n()) %>%
  group_by(variable1) %>%
  mutate(prop  = count/ sum(count))
```

- Group observations by using the `group_by()` function and include the following variables: `treated` and `performance<=0`. By executing this operation, we are is creating four groups: 1) Parties elected/above the threshold; 2) Parties elected/below the threshold; 3) Non elected parties/above the threshold; 4) Non elected parties/below the threshold.
- Add the `summarise()` function and create a new data frame from the grouping data frame we created before. Add `count` inside of the summarise function. This will be a variable that will store the number of observations in each group. For this, we need to add the `n()` function that calculates the number of observations in each group. By adding the two functions below, we get the number of observations in each of the four groups (feel free to run just these two lines and see what you get).
- Finally, let's calculate the *proportion* of compliers above and below the threshold. We can do this by grouping observations by their treatment condition and then using the `sum()` function. We can create a new variable called `prop`. It is based on the number of observations in each sub-group (whether they were above or below the threshold). Then, we divide this number by the total number of observations within each treatment condition using the `sum(count)` function. This will give you the proportion of compliance.

More detail of these function can be found below:

| Function/argument | Description |
|---|---|
| group_by() | Groups observations by the variables included in the function |
| summarise() | Creates a new data frame. It will have one (or more) rows for each combination of grouping variables |

| Function/argument | Description |
|---|---|
| n() | returns the number of observations in a current group |
| sum() | returns the sum of all the values present in its arguments |

*Reveal Answer*

```
enter %>%
  group_by(treated, performance <= 0) %>%
  summarize(count = n()) %>%
  group_by(treated) %>%
  mutate(prop  = count/ sum(count))
```

We can see the count and proportion of compliance in each group. We see that 176 times parties were below the threshold, but end up in parliament (around `10\%` of the observations). We can confirm that we have a case of one-sided non-compliance.

---

Let's subset the data for the following analyses. We want to make sure we have a fair comparison between observations below and above the cut-off. Since the `performance` variable cannot have outcomes `<-3,51`, we mirror the data by subsetting positive performance scores:

```
enter=enter[enter$performance<3.51,]
```

---

## Fuzzy RDD: Parametric Estimation

As last week, let's estimate a parametric fuzzy RDD first to make sure we understand the underlying logic - which is based on the instrumental variables design. Remember that in the case of *fuzzy RDDs* we have 'two events' of interest. First, **'treatment assignment'** which is defined by the running variable exceeding the cut-off, and, secondly, actual **treatment**

**status**. Let formally state that $Z_i$ is an indicator for when $X_{i,\text{centered}}$ exceeds the cut-off. We can then use this variable to instrument for the endogenous treatment variable $D_i$.

The principle is the same as in the IV design: We have an instrument, $Z_i$, that describes assignment to treatment based on an observation's position relative to the cut-off. We know that the instrument is a strong predictor - as it should be - of the treatment status, $D_i$. We can therefore exploit the random variation (close to the cut-off) introduced by the instrument to predict treatment status in our *first stage* - which you can think of as indicating compliance with the treatment assignment or describing the discontinuity in the probability of receiving treatment:

$$\hat{D}_i = \gamma_0 + \gamma_1 X_{i,centered} + \gamma_2 Z_i + \rho_i$$

for

$$c - h \leq X_i \leq c + h$$

Here, in a simple linear model with a single slope $\hat{D}$ predicts the treatment status based on assignment to the treatment, $(Z_i)$, and the vote share $(X_{i,centered})$ in $t_0$.

The second stage, then, uses the predicted values to calculate the following equation in order to estimate the LATE among compliers at the cut-off:

$$\hat{Y}_i = \beta_0 + \beta_1 X_{i,centered} + \beta_2 \hat{D}_i + \epsilon_i$$

for

$$c - h \leq X_i \leq c + h$$

Our coefficient of interest is $\beta_2$, which is the Local Average Treatment Effect ('LATE') among compliers at the cut-off. Having this in mind, we know we can estimate a parametric fuzzy RDD by specifying a 2SLS regression. This can be done using the packages and commands you already know, such as `iv_robust()`.

**Exercise 5: Estimate a parametric fuzzy RDD model using the `iv_robust()` command. Interpret your findings.**

*Reveal Answer*

```
model_fuzzy <- iv_robust(
  newsuccess ~ treated + performance | treat + performance,
  data = filter(enter)
)
summary(model_fuzzy)
```

Specifying the 2SLS model, we find that the estimated effect is `1.65%-points` on the *standardised scale*. The estimate is also statistically significant, so we conclude that the estimated LATE among compliers is meaningfully different from `0`.

Is this estimate robust though? The global parametric estimation has significant weaknesses. Think of our identification assumption for the RDD. In this global model, we have been using *all* observations from the data frame, including those rather far away from the threshold - i.e. those which are not really comparable. Moreover, the parametric model, unlike kernel estimators, does not weigh observations. Accordingly, the bias introduced by including the whole range of data is possibly severe. We might have estimated a biased effect here.

Let's tackle this problem by estimating a more credible parametric fuzzy RDD. We can do so by subsetting the data we input into the regression model.

**Exercise 6: Estimate the same model as above, but only consider parties with vote shares up to 1% on the standardised scale below/above the threshold. Is this result more credible?**

*Reveal Answer*

```
model_fuzzy_range <- iv_robust(
  newsuccess ~ treated + performance | treat + performance,
  data = filter(enter, performance <= 1, performance>-1)
)
summary(model_fuzzy_range)
```

This estimator draws a different picture. Our estimated effect is now `2.93%-points` on the standardised scale - this is meaningful and almost as large in magnitude as the standardised threshold itself. The estimate is also statistically significant. Based on this model specification, we can conclude that the LATE among compliers at the cut-off indeed is meaningful.

This model is somewhat more credible than the one we estimated above. Now we are only looking at observations which represent parties with a vote share of up to `1%-point` above or below the standardised threshold. This is more reasonable as as these parties should be more comparable. In fact, we find that wihtin this range the actual vote share (`performance`) does not predict the result of the subsequent election in any meaningful way. Within this rather

small window, entering parliament drives variation in the subsequent election.

Note that the model still has some disadvantages: We specify the functional form of the model and still do not weigh observations based on their distance to the cut-off. Estimating non-parametric models helps us calculate even more robust estimates.

---

## Non-Parametric Fuzzy RDD Estimation

In a first step, let's 'emulate' the parametric model we just specified - this time estimating a non-parametric model.

Fuzzy estimation with `rdrobust` is - thankfully - relatively straightforward. The syntax is essentially the same as in the case of sharp RDDs, obviously with the exception that we need to specify that our model is supposed to estimate a fuzzy RDD. We can do so using the `fuzzy=` argument to specify the treatment status (`treated` in our case). Importantly, you do not need to specify an instrument (or even create one!). All you need to specify is the column that indicates treatment status, the running variable and the cut-off — `rdrobust` will do all the above/below-the-cut-off instrument stuff behind the scenes for you.

**Exercise 7: Estimate a non-parametric fuzzy RDD with a bandwidth of `h=1` and uniform kernel.**

The syntax of `rdrobust` is pretty similar to the sharp RDD. Recall that the argument `c` specifies the cut-off, while `h` determines the bandwidth. The `fuzzy` argument is sufficient to have `rdrobust` estimate a fuzzy model.

```
summary(rdrobust(outcome, running_var, c=, fuzzy=treatment_status,   h=))
```

*Reveal Answer*

```
summary(rdrobust(enter$newsuccess, enter$performance, c=0, fuzzy=enter$treated,  kernel="u
```

How does that look like? The estimated effect is pretty similar to the one from our (narrow) parametric estimation. Conventional measures of uncertainty also indicate that the effect is significant - looking at the robust p-values and confidence intervals, however, reveals that we would *not* reject the null hypothesis based on the robust estimation - which we should base our conclusion on. Here, we find that there is no significant LATE among compliers.

As we know, `rdrobust`allows us to specify more appropriate models to estimate our treatment effect. Let's now make use of these opportunities to run more precise models.

**Exercise 8: Estimate the treatment effect, specifying a triangular kernel and a MSE-optimal bandwidth. Also, cluster standard errors by country using the `cluster=` argument.**

*Note: You can simply add the `cluster` argument and enter the relevant cluster variable. Recall that optimal bandwidths are specified using the `bwselect` argument instead of `h`.*

*Reveal Answer*

```
summary(rdrobust(enter$newsuccess, enter$performance, c=0, fuzzy=enter$treated,  kernel="t
```

We see that the MSE-optimal bandwidth corresponds to `1.13%-points` on the standardised scale on either side of the cut-off.

The estimated LATE among compliers is `1.86`, not that different from our previous point estimates. Under this model specification, we find that the estimated effect is not statistically significant - the robust confidence interval includes zero, so we cannot conclude that there indeed is a significant effect of entering parliament on parties' vote share in the subsequent election.

Note that this model uses a triangular kernel and accounts for non-independence among observations in the same country - for which reason it is somehwat more convincing compared to the models we estimated before.

We can also include covariates. The `rdrobust` package is very flexible in this regard and makes it easy to adjust for them. You can simply add the argument `covs=` and specify the

variables you want the model to include.

**Exercise 9: Specify a non-parametric fuzzy RDD with MSE-optimal bandwidth, clustered standard errors and `seats` and `newperc` as covariates. Interpret your result.**

*Reveal Answer*

```
summary(rdrobust(enter$newsuccess, enter$performance, c=0, fuzzy=enter$treated,  kernel="t
```

This again looks interesting - the optimal bandwidth hasn't changed much, nor did the magnitude of the estimated treatment effect. Similarly to the model we estimated in the exercise before, however, the treatment effect is not statistically significant to robust estimations.

---

## Sensitivity to Bandwidth

Can we be sure our estimated effect is robust to changing the bandwidth? Let's see if the estimated effect varies meaningfully. As last week, let's plot the effect over changing bandwidths.

**Exercise 10: Plot the estimated effect and 95% confidence intervals for the model specification from the previous exercise over bandwidths ranging from `0.1` to `3.5` in `0.1`%-points intervals.**

*Note: You migth want to use the following approach*

- *Create a data frame with all variables you need for the plot and a and observation for each bandwidth.*
- *Extract the values from the rdrobust output which you need for your plot.*
- *Loop the regression and the extraction of output over the bandwidths indicated in the initial data frame.*
- *Save the output in your loop to the respective row in the data frame.*
- *Plot the output from the newly created data frame.*

*Reveal Answer*

```
bandwidth <-  seq(from = 0.1, to = 3.5, by = 0.1)  #create a vector with values for each b

coefficient<- NA
se <- NA
obs <- NA
bw <- NA
ci_u <- NA
ci_l <- NA

data_extract <- data.frame(bandwidth, coefficient, se, obs, bw, ci_u, ci_l) # create a dat

# create a loop for each bandwidth that is indicated by 'i'
for(i in bandwidth){
  rdbw <- rdrobust(enter$newsuccess, enter$performance, c=0, fuzzy=enter$treated,  kernel=
  # extract the model output (make sure to extract *robust* statistics)
  data_extract$coefficient[data_extract$bandwidth==i]  <- rdbw$coef[3]
  data_extract$se[data_extract$bandwidth==i]  <- rdbw$se[3]
  data_extract$obs[data_extract$bandwidth==i]  <- (rdbw$N_h[1] + rdbw$N_h[2])
  data_extract$bw[data_extract$bandwidth==i]  <- (rdbw$bws[1, 1])
  data_extract$ci_l[data_extract$bandwidth==i]  <- rdbw$ci[3,1]
  data_extract$ci_u[data_extract$bandwidth==i]  <- rdbw$ci[3,2]
}

# Make sure the coefficient (and all other values) are numeric
data_extract$coefficient  <- as.numeric(data_extract$coefficient)

# Plot the estimates across bandwidths
ggplot(data = data_extract,
       aes(x = bandwidth, y = coefficient)) +
  geom_point(size = 0.8) +
  geom_ribbon(aes(ymin = ci_l, ymax = ci_u), alpha = 0.2) +
  geom_hline(aes(yintercept = 0), col = "red", linetype = 2) +
  coord_cartesian(ylim = c(-7.5, 7.5)) +
  theme_minimal() +
  labs(y = "LATE among Compliers at Discontinuity", x = "Bandwidths (Vote Margin)")
```

This looks interesting - the estimate varies vastly for very narrow bandwidths, but this is expected given the small number of observations that are close around the threshold. This

is not uncommon to see. However, it even looks like there is almost a significant *negative* treatment effect for bandwidths around `0.5%-points`. Our estimated *positive* treatment effect is robust to all bandwidths starting from around `2.6%` the end of the range. We can be confident that the effect is not contingent on specific bandwidths. What do we conclude?

Are we reasonably confident that there is a significant effect of entering parliament at $t_0$ on a party's vote share in the subsequent election ($t_1$)? What would be the effect in substantive terms though?

We know that the threshold (and parties' vote shares) have been standardised, with the standardised threshold being at `3.51%-points`. We can now write a function to convert the standardised (or synthetic) scale into substantive vote shares based on a country's electoral threshold:

```
standardised_to_vote <- function(coef, threshold) {
  vote_share <- coef*threshold/3.51
  return(vote_share)
}
```

**Exercise 11: Based on the treatment effect from our previous model from Exercise 9 (`2.057`), calculate the substantive treatment effect in Turkey, Greece and a country of your choice. For this exercise, let's forget that the estimated effect was not statistically significant - we just want to get an impression of its magnitude.**

*Reveal Answer*

```
# In Turkey, the threshold is 10%
standardised_to_vote(2.057, 10)

# In Greece, there's only a 3% threshold
standardised_to_vote(2.057, 3)
```

Recall that standardisation means that we are estimating the effect relative to the threshold. Accordingly, we will get different substantive results for countries with different absolute thresholds. The substantive effect will be roughly $\frac{6}{10}$ of the size of the threshold (because $2.057/3.51 \approx 0.6$).

We find that the substantive gain of entering parliament in Turkey corresponds to an increase of `5.86%-points` in the subsequent election, whereas it amounts to `1.76%-points` in Greece. What about the other country you chose?

---

## Falsification tests:

## Placebo outcomes:

We can conduct multiple tests to find evidence that there is no manipulation of the score around the cut-off. If units cannot precisely manipulate their scores, therefore we should expect no differences in terms of covariates for units above and below the threshold (we will test this later). We should also expect that there should not be differences in outcomes that should not be affected by the treatment. These outcomes are sometimes called 'pseudo-outcome' in the RDD literature.

One pseudo-outcome that we can use in this study is electoral success (standardised vote share) in the previous election ($t_{-1}$). The idea here is that current parliamentary representation in period ($t_0$) would not affect electoral performance in $t_{-1}$. Let's see if this is true.

**Exercise 12: Conduct a placebo outcome test using the placebo outcome variable `oldsuccess`. Use the `rdrobust()` function to perform this test. Set the argument `bwselect` equal to "mserd". Set the cut-off argument `c` equal to zero. Use the `summary()` command to report the results. What can you conclude based on this test? Does the evidence support the continuity assumption?**

*Reveal Answer*

```
summary(rdrobust(enter$oldsuccess, enter$performance, c=0, bwselect="mserd"))
```

We find that the local regression estimate is `-1.713` and the robust confidence interval goes from `-5.158` to `2.153` (and the p-value is `0.420`). We find that there no is evidence of manipulation of the running variable around the cut-off with respect ot previous electoral results.

## Placebo cut-offs

As we did last week, we can conduct a placebo cut-off test. This test implies looking at the outcome variable but using different cut-offs where we should not expect changes in the outcome. Thus, the estimates from this test should be near zero (and not statistically significant). Again, one **important** step to conduct this test is subsetting the sample for those observations that are above the cut-off and those that are below. We do this to avoid 'contamination' due to the real treatment effect. It also ensures that the analysis of each placebo cut-off is conducted using only observations with the same treatment status.

Rather than subsetting the data and creating a new data frame, let's use the `subset` argument in the `rdrobust()` function.

**Exercise 13: Conduct a placebo cut-off test using `-2.95` for the placebo tests below the cut-off. Use the `rdrobust()` function to perform this test. Set the argument `bwselect` equal to "mserd". Replace the cut-off argument `c` with the value of the placebo cut-off. Add the `subset` argument and set it equal to `data$performance < 0 & data$performance >-3.51`. Use `thesummary()`' command to report the results. What can you conclude based on this test? Does the evidence support the continuity assumption?**

*Reveal Answer*

```
summary(rdrobust(enter$newsuccess, enter$performance, c = -2.95, fuzzy=enter$treated, clus
                 bwselect="mserd", subset=enter$performance < 0 & enter$performance >= -3.
```

We find that the RD estimate is `-0.100`, which is pretty small. Also, the robust confidence interval reaches from `-17.186` to `14.109`, which clearly includes zero. Thus, this coefficient is not statistically significant. This provides evidence in favour of continuity assumption.

## Sorting

As we covered last week, the key identification assumption of RDDs is that there is no *sorting* on the running variable. That is, the running variable must be continuous around the threshold. We can do this using *density checks*. This means we're looking at the density of the running variable. Let's again use the `rddensity` package to do so.

**Exercise 14: Examine the density of the running variable using the `rddensity` command. Interpret your findings. See an example of the syntax below:**

Doing this is pretty straightforward. You only have to insert the running variable as an argument and, if different from 0, the cut-off. Note that you can optionally specify bandwidths, as above, using the `h` argument. Otherwise, an optimal bandwidth will be specified automatically. You can then save this as an object and run the `summary` command to see the output.

```
rdd <- rddensity(data$running_variable, c=0)
summary(rdd)
```

*Reveal Answer*

```
rdd <- rddensity(enter$performance, c=0)
summary(rdd)
```

This test reveals a p-value of `0.3239`. The null hypothesis of this test is that observations near the cut-off are allocated with a 0.5 probability. Given that we failed to reject the null hypothesis, we can claim there is no evidence of sorting around the cut-off.

Let's now plot this density test like we did last week. Let's again use the `rdplotdensity` function. Remember that this function uses the output from the `rddensity` command. The syntax is simple:

```
rdplotdensity([rdd_object] ,[running_var])
```

And these the arguments that you can use.

| Argument | Description |
|---|---|
| `plotRange` | Indicates starting and end point of plot |
| `plotN` | Number of grid points used on either side of the cut-off |
| `CIuniform` | True or False. If true, CI are displayed as continuous bands |

**Exercise 15: Use the `rplotdensity` to conduct a visual density check. Set the`plotRange` function equal to `c(-2,2)`. Set the `plotN` argument equal to `25`. Finally set the `CIuniform` equal to TRUE.**

*Reveal Answer*

```
plot_rdd <- rdplotdensity(rdd, enter$performance, plotRange = c(-2, 2), plotN = 25,  CIuni
```

Again, we find visual evidence for continuity around the cut-off. Now we can be confident about sorting not being a problem in this data.

You can also conduct the same test using the `DCdensity` function from the `rdd` package. Unfortunately, to use this command, you need to drop the observations with missing values. You can use `drop_na(running_variable)` function to do this.

## Balance

Let's finally conduct a balance test to examine whether near the cut-off treated units are similar to control units in terms of observable characteristics. Ideally, we should get an RD estimate, $\tau_{RD}$, which is equal to zero.

**Exercise 16: Let's check for balance around the cut-off. Use the `rdrobust()` function. Set the cut-off argument `c` equal to 0, and the `bwselect` argument equal to "mserd". In this case, the outcome variables are the covariates that we want to check for balance. Check if there is balance for the covariate `oldparty`, which is the 'age' of the party in terms of previously contested elections. Remember to use the `summary()` command.**

*Reveal Answer*

```
summary(rdrobust(enter$oldparty, enter$performance, c=0, bwselect="mserd"))
```

We observe that there is balance for the covariates as it should be.