Profesor: Héctor Bahamonde, PhD.

e:hector.bahamonde@uoh.cl

w:www.hectorbahamonde.com

Curso: MLE.

TA: Gonzalo Barría.

Regression Discontinuity Designs (RDDs)—Fuzzy Designs

Motivemos el modelo RDD general:

$$y_i = \beta_0 + \beta_1 x_i + \pi z_i + \epsilon_i \tag{1}$$

En los diseños RDD *sharp* existe una relación determinista en como z clasifica a x (Hahn, Todd, and Klaauw 2001, p. 202):

$$z_i = \begin{cases} 1 & \text{si } x \leq \text{corte} \\ 0 & \text{si } x > \text{corte} \end{cases}$$

Ya sabemos que el corte no es aleatorio. En los diseños *fuzzy* z no es determinista, si no que hay variables adicionales que son anteriores a z que operan en la clasificación de x.

El problema substantivo detrás del z en un *fuzzy design* es un problema de "*compliance*": no todos los sujetos involucrados en el diseño "cumplen" con su régimen experimental esperado. Por ejemplo, los sujetos que debieran haber tomado el tratamiento, no lo hicieron (el experimento salió mal!).

O también, el experimentador no tenía las herramientas para "forzar" la clasificación experimental. Imagínate que estudiando campañas políticas envías por carta un tríptico con información sobre una campaña. Después, la idea es llamar por teléfono a hogares de control (sin tríptico) y tratamiento (con tríptico) y hacerles una encuesta. *Qué pasa si aquellos que debieran haber sido tratados no leyeron el tríptico?* Esas personas debieran haber sido "tratadas", pero dado que no leyeron el folleto, ahora están en el régimen de control. Aquí el experimentador no puede forzar la lectura del folleto.

Otro tipo de diseños fuzzy *no* tiene que ver con errores en el proceso de diseño experimental. Por ejemplo, Meyersson (2014, p. 244) se interesó en si el grado en que los partidos políticos islámicos afectaba el empoderamiento femenino. Aquí el eje x es el margen por el que los partidos islámicos

ganan las elecciones, y el eje y es la participación de mujeres que asiste al colegio. En este caso es fácil ver que existirá un *overlap* entre las observaciones, no existiendo una clasificación determinista entre quienes recibe el "tratamiento" y quienes no.

Entonces, *fuzzy* va más allá de errores en la administración del tratamiento.

Existen dos maneras de analizar este tipo de datos:

1. Instrumental Variables y 2SLS.
2. Regresión Lineal Local ("*local linear regression*").

I. Estimador

En los diseños *sharp* trabajamos con el ATE o ATT. Sin embargo, en los diseños *fuzzy* no existe "*perfect compliance*". Es por esto que debemos trabajar con otro estimador, el *Intention-to-treat effects* o *ITT*. Este estimador no se concentra en quien recibió el tratamiento, sino en quien nosotros "*intentábamos*" tratar.

El ITT es más sencillo y realista. En general se entiende que el ITT es la diferencia entre los sujetos y_1 bajo régimen de tratamiento $z(1)$ y los sujetos y_0 bajo régimen de control $z(0)$, independiente de problemas "*compliance*" y otros posibles problemas que puedan existir. Al obviar todas las posibles alternativas que podrían haber salido mal (*compliance*, por ej.), el ITT mide la *elegibilidad* al tratamiento (z_1), independiente de si se trata de y_1 o y_0.

II. Instrumental Variables y 2SLS

Dado que z asigna 0's o 1's con cierta *probabilidad* (i.e. no *deterministamente*), el "no-cumplimiento" es posible de ser modelado. De hecho, Hahn, Todd, and Klaauw (2001) demuestran que z puede ser estimado de la misma manera que una **variable instrumental** vía a *two-stage least square design* (2SLS).

La manera en que hacemos esto es que estimamos z como función de x en la primera etapa. Después, se obtiene $\hat{z}$. En la segunda etapa se estima y como función de x y $\hat{z}$.

Es decir, en la primera etapa se estima Equation 2

$$z_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{2}$$

y en la segunda etapa se estima Equation 3

$$y_i = \beta_0 + \beta_1 x_i + \pi \hat{z}_i + \epsilon_i \tag{3}$$

Cocinemos unos datos. En esta parte usaremos el paquete `rddtools` que se encarga de estimar el modelo vía 2SLS.

```
mu <- c(0, 0) # los promedios de ambos vectores
sigma <- matrix(c(1, 0.7, 0.7, 1), ncol = 2) # una matriz de varianza-covarianza
p_load(MASS)
d <- as.data.frame(mvrnorm(2000, mu, sigma)) # creamos la base de datos
colnames(d) <- c("x", "y") # le cambiamos los nombres
corte = 0 # definir el corte
d$z <- ifelse(d$x <  corte, 0, 0.8) # introducimos FUZZYNESS
# Si d$x < corte, z es 0 con un 100% de probabilidad.
# Si d$x > corte, z es 1 con un 80% de probabilidad.
# "Compliance"!
```

Aquí hemos cocinado datos multivariados usando la función `mvrnorm` (*multivariate random normal*). Como es multivariada, tendremos más de un vector (en este caso dos).

Hasta ahora `d$z` es un vector de probabilidades (0, y .8).

```
table(d$z)

##
##    0  0.8
##  990 1010
```

Metamos ese vector de probabilidades en una función para convertirlos en 0's y 1's

```
d$z <- sapply(X = d$z, FUN = function(x) rbinom(1, 1, prob = x)) # funcion
```

Veámos cómo quedó:

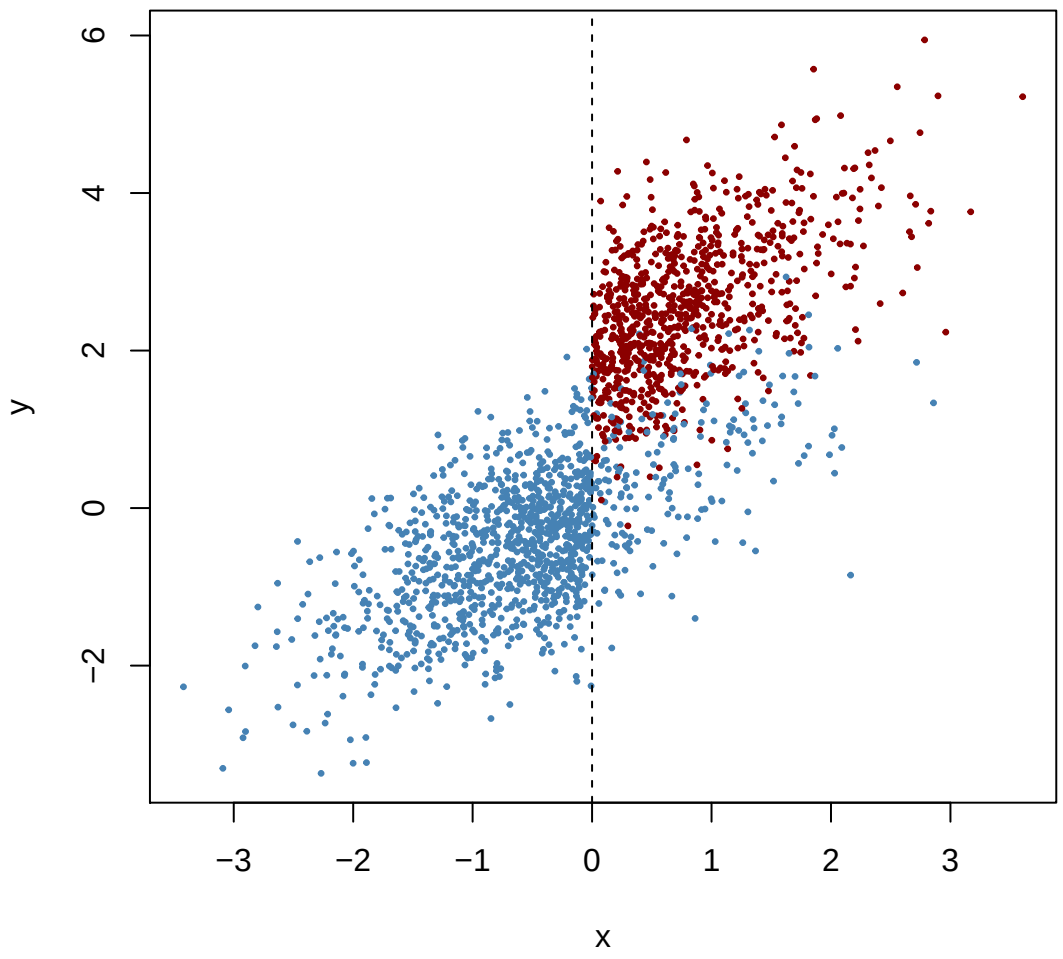
```
table(d$z)

##
##    0    1
## 1177  823
```

Ahora fabriquemos el efecto de z en y, y digamos que es 2 (combinación lineal).

```
d$y <- d$y + d$z * 2
```

Ahora grafiquémos

```
plot(d$x, d$y,
     col = c("steelblue", "darkred")[factor(d$z)],  # asignar color para z(1) y z(0)
     pch= 20, # geometria de los circulos
     cex = 0.5, # tamano de los circulos
     xlab = "x",
     ylab = "y")
# linea vertical
abline(v = 0, lty = 2)
```



Ahora que tenemos los datos listos, estimemos el modelo.

```
# Estimar Fuzzy RDD
p_load(rddtools)
# declarar que objeto "d" es un data frame de RDD
data <- rdd_data(d$y, d$x,
                 cutpoint = corte,
```

```
                z = d$z
                )
# veamos
head(data)

##            x         y z
## 1  0.7605976  1.934505 1
## 2  0.3326682  2.223360 1
## 3 -1.1249196 -0.899735 0
## 4 -1.4135165 -0.670849 0
## 5 -2.6293837 -2.527171 0
## 6  1.2715528  2.057119 1

# Estimar modelo fuzzy RDD
rdd_reg_lm(rdd_object = data, slope = "same")

## ### RDD regression: parametric ###
##  Polynomial order:  1
##  Slopes:  same
##  Number of obs: 2000 (left: 990, right: 1010)
##
##  Coefficient:
##   Estimate Std. Error t value             Pr(>|t|)
## D 1.905460   0.067511  28.224 < 0.00000000000000022 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# En este caso, D es el efecto causal, o rho. Es aprox. 2.
```

III. *Local Linear Regression*

Otra manera de estimar es estimando un modelo justo en el corte. Este es un enfoque que usa sólo las observaciones cerca del corte—i.e. una regresión *local*—en una especie de "mini regresión".

- El problema es que perdemos poder estadistico (?).
- El supuesto es que las observaciones que rodean el corte son muy similares en cuanto a sus valores de x. Por que es esto importante? En base a este supuesto se construye el *contrafactual*.

Si quisiéramos estimar una regresión local, la primera pregunta sería cuán ancha o delgado es nuestro "ancho de banda" (o "*bandwidth*"). Afortunadamente Imbens and Kalyanaraman (2012) desarrollaron un estimador que encuentra el "ancho de banda" más óptimo al rededor del corte. Ellos demuestran que usando sólo esas observaciones al rededor del corte se minimizan los errores cuadrados del ("mini") modelo. Este estimador está contenido en la función `IKbandwidth` del paquete `rdd`.

```
p_load(rdd)
band <- IKbandwidth(d$x, d$y)
band # ancho de banda optimo

## [1] 0.4034962

# corta la base segun el ancho de banda optimo.
# nos quedamos solo con los valores determinados por ``band'' por lado
d_rdd  <- d[d$x < band & d$x > -band, ]
# veamos la max y min de x de la nueva base datos cortada
summary(d_rdd) # base cortada

##        x                   y                z
##  Min.   :-0.403293   Min.   :-2.2574   Min.   :0.0
##  1st Qu.:-0.180701   1st Qu.:-0.3422   1st Qu.:0.0
##  Median : 0.006590   Median : 0.4866   Median :0.0
##  Mean   : 0.006631   Mean   : 0.7247   Mean   :0.4
##  3rd Qu.: 0.214356   3rd Qu.: 1.8554   3rd Qu.:1.0
```

```
##  Max.   : 0.401601   Max.   : 4.2753   Max.   :1.0

# estimamos el modelo (la mini regres.) con la base cortada d_rdd
linear_rdd_model <- lm(y ~ x + z, data=d_rdd)
summary(linear_rdd_model) # z es el efecto cuasi causal local del ``tratamiento''

##
## Call:
## lm(formula = y ~ x + z, data = d_rdd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.39135 -0.48548 -0.01219  0.47551  2.20708
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept) -0.02193    0.04333  -0.506                0.613
## x            1.12980    0.17876   6.320       0.000000000489 ***
## z            1.84796    0.08318  22.217 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7296 on 642 degrees of freedom
## Multiple R-squared:  0.6966,Adjusted R-squared:  0.6957
## F-statistic: 737.2 on 2 and 642 DF,  p-value: < 0.00000000000000022
```

En este caso, `IKbandwidth` estima que debemos seleccionar valores en x entre 0.4034962 a la izquierda del corte, y 0.4034962 a la derecha del corte. Es decir, si el corte es 10, debemos hacer una nueva base de datos cuyos valores de x estén comprendidos entre $10 - 0.4034962 = 9.5965038$ ("para la izquierda") y $10 + 0.4034962 = 10.4034962$ ("para la derecha").

```
knitr::purl('RDD_Fuzzy.Rnw')

## [1] "RDD_Fuzzy.R"

Stangle('RDD_Fuzzy.Rnw')

## Writing to file RDD_Fuzzy.R
```

References

Hahn, Jinyong, Petra Todd, and Wilbert Klaauw (Jan. 2001). "Identification and Estimation of Treatment Effects with a Regression-Discontinuity Design." In: *Econometrica* 69.1, pp. 201–209.

Imbens, Guido and Karthik Kalyanaraman (July 2012). "Optimal Bandwidth Choice for the Regression Discontinuity Estimator." In: *The Review of Economic Studies* 79.3, pp. 933–959.

Meyersson, Erik (2014). "Islamic Rule and the Empowerment of the Poor and Pious." In: *Econometrica* 82.1, pp. 229–269.