Introducción a métodos econométricos en R

27 de septiembre de 2021

INSTITUTO TECNOLOGICO AUTONOMO DE MEXICO (ITAM)

Seminario de Investigación Económica

Instructor: Horacio Larreguy

Asistente: Manuel Quintero

1. Introducción a R y RStudio

De acuerdo con el sitio oficial de R: *The R Fundation (2021)*. R es un lenguaje de programación de **software libre** para la computación y gráficos estadísticos. El lenguaje R incluye una amplía variedad de técnicas y pruebas estadísticas para modelos tanto lineales como no lineales.
Fortalezas de R:

- Facilidad en el manejo de datos de bases pequeñas y medianas ($<$ 2 GB).
- Conjunto de operadores para el calculo de matrices.
- Variedad de paquetes bien documentados para el análisis estadístico.

Debilidades de R:

- Lee todo los datos a la memoria RAM desde un inicio (no los fragmenta).
- Límite aproximado de 2 a 4 GB de almacenamiento o un índice vectorial de 2 mil millones.
- Todos los objetos de R se almacenan en la memoria, hay que eliminarlos constantemente.

RStudio es un entorno de desarrollo integrado (IDE) tanto para R como para Python que cuenta con un editor, una consola, ambiente global de trabajo, visualización de gráficas en una interfaz gráfica de usuario (GUI). Además, se puede utilizar para trabajar con Markdown (RMarkdown), HTML, LaTeX, entre otros lenguajes de programación.

Instalación: Las versiones más recientes de R y RStudio se pueden descargar desde los siguientes enlaces:

1. R: https://cran.r-project.org/bin/windows/base/.
2. RStudio: https://www.rstudio.com/products/rstudio/.

1.1. Básico de RStudio

El libro del *R Development Core Team (2000)* escrito por los desarrolladores contiene una introducción detallada al lenguaje de programación R.

R, al ser un *software* libre, gran cantidad de desarrolladores y científicos crean sus paquetes para hacer que sus investigaciones sean reproducibles; por lo general, estos paquetes contienen funciones para realizar una tarea en especifica, muchos de estos paquetes son aceptados por la comunidad científica para su uso en trabajos de investigación. Estos paquetes son de gran ayuda para obtener resultados rápidamente, sin tener que repetir el trabajo ya realizado por otros. Por esto, para empezar a trabajar en R es necesario saber como instalar y cargar paquetes, así como conocer otros comandos básicos.

Limpiar el *Global Environment* o *workspace*

```
rm(list = ls()) # Limpia todo lo almacenado en el workspace
rm(list = setdiff(ls(), "x")) # Elimina todo excepto la variable x

.rs.restartR() # Reiniciar la sesión: limpia el workspace y paquetes
```

Comando para instalar un paquete o actualizarlo

```
install.packages("nombre")
# Podemos utilizar un paquete sin instalarlo de la siguiente forma,
nombreDelPaquete::funcion # Donde función es una operación dentro del paquete
    nombreDelPaquete
```

Comando para desinstalar un paquete

```
remove.packages("nombre")
```

Comando para cargar un paquete y poder utilizarlo

```
library(nombre)
```

Comando para cambiar directorio de trabajo [1]

[1] Véase el Apéndice A para un ejemplo de código para limpiar el directorio, instalar y cargar múlti-

```r
setwd("C:/Users/nombre/Desktop/Tutorial") # establece directorio
getwd('') # string que regresa el directorio actual
```

Comando para obtener ayuda de una función específica

```r
help(mean)
?mean
```

Vectores:

```r
# Vectores
palabras < -c("carro", "barco", "avión")
x <- c(1,2,3,4)
y <- c(2,4,6,8)

# Aritmética vectorial
sqrt(x) # Raiz cuadrada
x + y # Suma
2*x*y # Producto
x/y  # Division entrada a entrada

length(x) # Longitud del vector
sum(x) # Suma las entradas del vector
```

Sucesiones

```r
# Generar sucesiones
x <- c(1:10) # Ascendente del 1 al 10, enteros
y <- c(10:1) # Descendente del 10 al 1, enteros
z <- seq(0, 10, by=.25) # Del 0 al 10, incrementos en 0.25
w <- rep(c(1,2),2) # Dos vectores c(1,2) concatenados
```

Operadores lógicos

```r
x > y # TRUE si x es mayor que y
x >= y # TRUE si x es mayor o igual a y
x < y # TRUE si x es menor que y
x <= y # TRUE si x es menor o igual a y
x == y # TRUE si x es igual a y
x != y # TRUE si x diferente de y
is.na(x) # TRUE si x es NA o NaN
!is.na(x) # TRUE si x no es NA y no es NaN
x == 1 & y == 2 # Intersección
```

ples paquetes al mismo tiempo y establecer directorio de trabajo donde se encuentra el archivo.

```
x == 1 | y == 2 # Unión
```

2. Datos en R

2.1. Lectura de datos

Importación e exportación de archivos .txt (read.table())

```
read.table('name.txt', header = F, sep = "") # Importar
write.table(datos, file = '', sep = " ", row.names = F, col.names = T) # Exportar
```

Importación e exportación de archivos .xlsx o .xls (paquetes xlsx o readxl)

```
library(xlsx)
read_excel('path', sheet = NULL, col_names = T, ...) # Determina automáticamente si es
    xls o xlsx
read_xlsx('path.xlsx', sheet = NULL, col_names = T, ...)
read_xls('path.xls', sheet = NULL, col_names = T, ...)

write.xlsx(data, file = '', sheetName = "name") # Exportar
```

Importación e exportación de archivos .csv (read.csv())

```
read.csv('file.csv', header = T, sep = ",", encoding = 'UTF-8', ...) # Importar
write.csv(data, file = '' , row.names = F, col.names = T, fileEncoding = 'UTF-8') #
    Exportar
```

Importación e exportación de archivos .dta (paquetes haven o readstata13)

```
library(haven)
read_dta('file.dta', encoding = '', skip = '')
write_dta(data, path = '')

# readstata13 es mejor para nombres de variables mayores a 32 caracteres
library(readstata13)
readstata13(data, )
save.dta13(data, file = '')
```

Importación e exportación de archivos .shp o .geojson (paquete sf)

```
st_read(dsn, layer, ...) # Importar
st_write(obj, dsn, layer, ...) # Exportar
```

2.2. Manejo de datos

La paquetería clásica para el manejo de datos es dplyr y haciendo uso del operador *Pipe*: %>%. El paquete tidyverse es el paquete madre de dplyr.

Generamos 2 bases de datos aleatorias

```
# Generamos números aleatorios de una distribución exponencial
datos_x <- cbind.data.frame(floor(matrix(rexp(25, rate=.1), ncol=5)), c(1:5))
names(datos_x) <- c(paste0("X", c(1:5)), "id")

# Generamos números aleatorios de una distribución normal
datos_y <- cbind.data.frame(matrix(rnorm(50), ncol=5), c(1:10))
names(datos_y) <- c(paste0("Y", c(1:5)), "id")
```

Instalamos el paquete

```
install.packages("tidyverse")
library(dplyr)
```

Distintas funciones de dplyr para el manejo de datos

```
# Comando para seleccionar columnas
select(datos_x, -4)
# Comando para crear nueva columna
mutate(datos_x, X8 = X1 - X2 + X3)
# Comando para eliminar renglones
slice(datos_x, -n()) # Elimina el ultimo renglón
# Comando para fusionar 2 columnas con valores NA
mutate(datos_x, NUEVA = coalesce(X6,X7))
# Comando para filtrar por ciertos valores
filter(datos_x, X6 == 1)
# Comando para renombrar columnas
rename(datos_x, EDAD = X2)
```

Utilizando el operador %>%.

```
# Hacer todo junto
nueva_base_x <- datos_x %>% dplyr::select(c(-4)) %>% mutate(X8 = X1 - X2 + X3) %>%
    slice(-n()) %>%
  mutate(NUEVA = coalesce(X6,X7)) %>% filter(X6 == 1) %>% rename(EDAD = X2)
```

Colapsar datos desagregados

```
# Agrupar por variables de interés
group_by(datos_x, nombre_Variable)

# Colapasar los datos a nivel nombre_Variable
summarise_all(datos_x, sum, na.rm = T) # sum es la función de la forma que queremos se
     colapsen los datos.

# Agregar datos a nivel SECCION y ENTIDAD
Base_de_datos <- Base_de_datos %>% select(c(indices o nombres de las variables que te
    interesan)) %>% group_by(ENTIDAD, SECCION) %>% summarise_all(sum, na.rm = TRUE)
```

Reshape

Una practica común al trabajar con bases de datos es la transformación o reordenamiento de la misma. Las operaciones más comunes son permutar renglones y columnas, resumir y transponer los datos. En R, se puede hacer uso del paquete reshape2 para realizar este tipo de operaciones. Puede encontrar una explicación más detallada de las siguientes funciones en el tutorial de Datacamp: Smith (2020).

- cbind() y cbind.data.frame() son funciones que sirven para combinar vectores, matrices y bases de datos por columnas.
- rbind() y rbind.data.frame() son funciones que sirven para combinar vectores, matrices y bases de datos por renglones.
- t() es la función para obtener la transpuesta de un vector, una matriz o una base de datos.
- melt() selecciona múltiples columnas de una base y las convierte en una sola columna.
- dcast() revierte lo que hace la función melt().

Fusionar bases de datos[2]

- inner_join(): retiene solo los renglones de x que tienen pareja en y y todas las columnas de x e y.
- left_join(): retiene todas las columnas de x e y y los renglones de x que no tienen pareja en y, tendrán valores NA en las nuevas columnas. Si hay múltiples parejas, todas las combinaciones se regresan.
- right_join(): retiene todas las columnas de x e y y los renglones de y que no tienen pareja en x, tendrán valores NA en las nuevas columnas. Si hay múltiples parejas, todas las combinaciones se regresan.
- full_join(): retiene todos los valores de x e y con NA donde no hay pareja.

```
inner_join(datos_x, datos_y, by = "id")
left_join(datos_x, datos_y, by = "id")
right_join(datos_x, datos_y, by = "id")
full_join(datos_x, datos_y, by = c("idx" = "idy")) # Si la variable del merge
    tiene distintos nombres
```

[2]También son útiles las funciones semijoin(), antijoin() nestjoin(). Para ver una descripción más detallada de las siguientes funciones, véase **tidy**.

3. Regresión lineal

La función clásica para estimar modelos de regresión lineal en R por el método de mínimos cuadrados ordinarios es lm(), *linear models*, la cual viene como paquetería base de R.

```
lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE,
    y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)
```

El argumento formula es una expresión de la forma $Y \sim X$, donde Y es la variable dependiente y X la matriz de variables explicativas. El segundo argumento, *data* especifica cual es la base de datos que contiene las variables que aparecen en formula.

En esta sección trabajaremos con la base datos **iris** que cuenta con 150 observaciones y 5 variables de una planta conocida como iris.

```
data(iris) # Cargamos la base de datos iris de R
names(iris) # Vemos el nombre de las variables en la base de datos
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Corremos una regresión lineal simple tomando como variable dependiente el largo del sépalo de la planta y como variable explicativa el ancho del sépalo de la planta. Creamos el modelo usando el comando **lm()** y obtenemos el resumen de los resultados con la función **summary()**.

```
# Creamos un modelo de regresión lineal simple
reg_simple <- lm(Sepal.Length ~ Sepal.Width, data = iris)
# Imprimimos los resultados
summary(reg_simple)

Call:
lm(formula = iris$Sepal.Length ~ iris$Sepal.Width, data = iris)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5561 -0.6333 -0.1120  0.5579  2.2226

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)        6.5262     0.4789   13.63   <2e-16 ***
iris$Sepal.Width  -0.2234     0.1551   -1.44    0.152
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1         1

Residual standard error: 0.8251 on 148 degrees of freedom
Multiple R-squared:  0.01382, Adjusted R-squared:  0.007159
F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

Una regresión lineal múltiple se corre de manera análoga, escribiendo el argumento formula de la siguiente manera:

$$Y \sim X_1 + X_2 + ... + X_n,$$

Por ejemplo, explicamos el largo del sépalo por la anchura tanto del sépalo como del pétalo.

```
# Regresión lineal múltiple
reg_multiple1 <- lm(Sepal.Length ~ Sepal.Width + Petal.Width, data = iris)
reg_multiple2 <- lm(Petal.Length ~ Sepal.Width + Petal.Width + Sepal.Length, data =
    iris)
```

Cómo implementar efectos fijos: una alternativa para implementar efectos fijos es hacer uso de la función felm del paquete lfe.

Supongamos que queremos introducir efectos fijos por el tipo de especie de la planta:

```r
# Para incluir efectos fijos utilizamos factor()
regresion_ef <- lm(Sepal.Length ~ Sepal.Width + factor(Species), data = iris)

# La mejor alternativa a estos modelos es hacer uso de felm del paquete lfe.
require(lfe)
regresion_ef_felm <- felm(Sepal.Length ~ Sepal.Width | Species, data = iris)
```

4. De R a LaTeX

Cuando queremos reportar los resultados de una manera elegante y formal tenemos que exportar los resultados de la función **summary()** en una mejor presentación. En R, tenemos dos maneras de trabajar con LaTeX. La primera es trabajar directamente en **R Markdown**, que nos permite escribir reportes que contengan código R y compilen un documento en formato .pdf, .html o .doc interpretando correctamente comandos HTML o LaTeX. Usualmente, solo queremos generar el código en TeX de ciertos resultados para importarlos a un compilador de LaTeX. Los paquetes más utilizados para exportar resultados econométricos a tablas en código TeX son xtable y stargazer.

4.1. Paquete xtable

Utilizando la función xtable podemos presentar en LaTeXlos resultados obtenidos por la función **summary()**.

```
xtable(x, caption = NULL, label = NULL, align = NULL, digits = NULL,
       display = NULL, auto = FALSE, ...)
```

Por ejemplo, para obtener el código TeX de la regresión lineal simple presentada en la sección anterior, utilizamos el siguiente código:

```
# Tabla de resumen de resultados
library(xtable) # Cargamos el paquete xtable

print(xtable(reg_simple, caption = 'Resultados de la regresion lineal simple'),
    caption.placement = 'top')
```

Tabla 1: Resultados de la regresión lineal simple

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	6.5262	0.4789	13.63	0.0000
Sepal.Width	-0.2234	0.1551	-1.44	0.1519

Note que estamos utilizando la función print() para modificar más atributos de la tabla del objeto *table* de LaTeX.

4.2. Paquete stargazer

El **stargazer** fue creado por Marek Hlavac de la universidad de Harvard y es uno de los más desarrollados para exportar resultados estadísticos de R a LaTeXcon la función stargazer(). Esta función tiene una gran variedad de opciones para personalizar las tablas tanto en formato HTML como en formato TeX. La documentación del paquete se puede consultar aquí.

```
# Tabla sencilla para una regresión lineal simple
stargazer(reg_simple) # Por default el resultado es código "latex"
```

Además, con stargazer podemos presentar múltiples regresiones en una sola tabla. Si las especificaciones comparten variables explicativas, estás se alinean automáticamente en renglones.

También podemos especificar que estadísticos incluir, como el estadístico F, R, R ajustado, el número de observaciones entre otros, con el atributo *omit.stat*. Un atributo muy útil es *add.lines* que lo utilizamos para agregar renglones con información adicional en la parte inferior de la tabla.

```
# Calculamos el valor p de la prueba F con el paquete car
library(car)
pval_f1 <- linearHypothesis(reg_multiple1, c("Sepal.Width", "Petal.Width"), c(0,0),
    test = "F")[6]$`Pr(>F)`[2]
pval_f2 <- linearHypothesis(reg_multiple2, c("Sepal.Width", "Petal.Width", "Sepal.
    Length"), c(0,0), test = "F")[6]$`Pr(>F)`[2]

# Dos regresiones multivariadas en una sola tabla
tabla2 <- stargazer(reg_multiple1, reg_multiple2,
          header = FALSE,
          font.size = "scriptsize",
          dep.var.labels.include = FALSE,
          table.placement = "H",
          column.labels = c("Largo del Sépalo", "Largo del pétalo"),
```

```
          covariate.labels = c("Ancho del Sépalo", "Ancho del pétalo",  "Largo del pé
              talo", "Constante"),
          omit.stat = c("f", "ser","adj.rsq"),
          add.lines = list(c("Prueba F: \\textbf{T} $=$ 0 (valor p)", round(pval_f1,4)
              , round(pval_f2))),
          title = "Ejemplo de múltiples regresiones multivariadas en una sola tabla",
          type = "latex")
# Un problema con stargazer es la forma forma de agregar las notas, este que se
    presenta aquí es una forma fácil.
note.latex <- "\\multicolumn{3}{l} {\\parbox[t]{6cm}{ \\textit{Notas:}
De esta manera podemos modificar las notas. * denota p$<$0.1, ** denota p$<$0.05, y **
    * denota p$<$0.01.}} \\\\"
tabla2[grepl("Note", tabla2)] <- note.latex
# Imprimimos el código de LaTeX guardado en el objeto tabla2
cat(tabla2)
```

Tabla 2: Ejemplo de múltiples regresiones multivariadas en una sola tabla

	Variables dependientes:	
	Largo del Sépalo	Largo del pétalo
	(1)	(2)
Ancho del Sépalo	0.399***	−0.646***
	(0.091)	(0.068)
Ancho del pétalo	0.972***	1.447***
	(0.052)	(0.068)
Largo del pétalo		0.729***
		(0.058)
Constante	3.457***	−0.263
	(0.309)	(0.297)
Prueba F: $\mathbf{T} = 0$ (valor p)	0	0
Método: MCO	✓	✓
Observaciones	150	150
R^2	0.707	0.968

Nota: De esta manera podemos modificar las notas. * denota p<0.1, ** denota p<0.05, y *** denota p<0.01.

5. Diferencias en diferencias

Buscamos estimar el siguiente modelo:

$$Y_{it} = \beta_0 + \beta_1 T_i + \beta_2 \text{post}_t + \delta T_i \cdot \text{post}_t + \epsilon_{it},$$

donde, Y_{it} es la variable endógena, T_i y post_i son indicadoras de tratamiento y periodo, respectivamente. El parámetro δ es el verdadero efecto en el tratamiento o la diferencia en diferencia y ϵ_i es el error aleatorio.

5.1. Ejemplo 2 periodos

A manera de ilustración consideremos el siguiente escenario. Una empresa realiza un programa de asesoramiento de empleo a 20 trabajadores que habían egresado recientemente de la universidad, al inicio se les preguntó cual era el salario que percibían por hora en dólares. Posteriormente, en dos grupos de 10 personas se implementó el tratamiento: asesoramiento para encontrar empleos mejor pagados. Al grupo 1 se le mostró una corta presentación sobre como prepararse mejor para las entrevistas laborales y al segundo grupo se le acompañó durante el proceso de aplicación a una nueva empresa. Los dos grupos tenían que aplicar a un nuevo empleo al finalizar el tratamiento y reportar el salario ofrecido.

```
# Generamos datos aleatoriamente
set.seed(32) # Semilla aleatoria
# Un vector de 10 enteros generados aleatoriamente entre 10 y 20
wage1 <- floor(runif(10, min=10, max=20))
# Un vector de 10 enteros generados eleatoriamente entre 11 y 21
wage_post_1 <- floor(runif(5, min=11, max=21))
# Un vector de 10 enteros generados eleatoriamente entre 20 y 30
wage2_post_2 <- floor(runif(5, min=20, max=30))
# Concatemaos el vector de salarios
wage <- c(wage1, wage_post_1, wage2_post_2)
# Variable indicadora despues de tratamiento
post <- c(rep(0,10), rep(1,10))
# Indicadora de tratamiento
```

```
treatment <- c(rep(1,5), rep(0,5), rep(1,5), rep(0,5))
dd_data <- cbind.data.frame(wage, post, treatment) # Datos
colnames(dd_data) <- c("wage", "post", "Treatment")
```

Creamos un modelo de Diferencias en Diferencias (DD) en R utilizando la función **lm()**.

```
# Corremos la regresión DD
did_reg <- lm(wage ~ post*Treatment + post + Treatment, data = dd_data, singular.ok =
    TRUE)

# Tabla de los resultados
stargazer(did_reg,
          header = FALSE,
          font.size = "scriptsize",
          dep.var.labels.include = FALSE,
          table.placement = "H",
          column.labels = "Salario",
          covariate.labels = c("Post", "Tratamiento (Grupo 2)", "Post x Tratamiento",
              "Constante") ,
          omit.stat = c("f", "ser","adj.rsq"),
          add.lines = list(c("Outcome range", "[10, 30]")),
          title = "Diferencias en diferencias en el salario de los trabajadores
              inscritos en el programa",
          type = "latex")
```

Tabla 3: Diferencias en diferencias en el salario de los trabajadores inscritos en el programa

	Salario
Post	0.600
	(1.432)
Tratamiento (Grupo 2)	1.400
	(1.432)
Post x Tratamiento	9.400***
	(2.025)
Constante	15.200***
	(1.012)
Rango DV	[10, 30]
Observaciones	20
R^2	0.842
Notas:	*p<0.1; **p<0.05; ***p<0.01

Recordemos que en una regresión DD podemos obtener las medias de los distintos grupos antes y después del tratamiento de la siguiente forma:

$$\begin{aligned}
\mathbf{E}[\text{Group 1} \mid \text{post} = 0] &= \beta_0, \\
\mathbf{E}[\text{Group 2} \mid \text{post} = 0] &= \beta_0 + \beta_1, \\
\mathbf{E}[\text{Group 1} \mid \text{post} = 1] &= \beta_0 + \beta_2, \\
\mathbf{E}[\text{Group 2} \mid \text{post} = 1] &= \beta_0 + \beta_1 + \beta_2 + \delta.
\end{aligned}$$

En R extraemos los coeficientes (o suma de coeficientes) de la siguiente manera:

```
did_reg$coefficients[1] # beta_0
did_reg$coefficients[1] + did_reg$coefficients[2] # beta_0 + beta_1
did_reg$coefficients[1] + did_reg$coefficients[3] # beta_0 + beta_2
sum(did_reg$coefficients) # beta_0 + beta_1 + beta_2 + delta
```

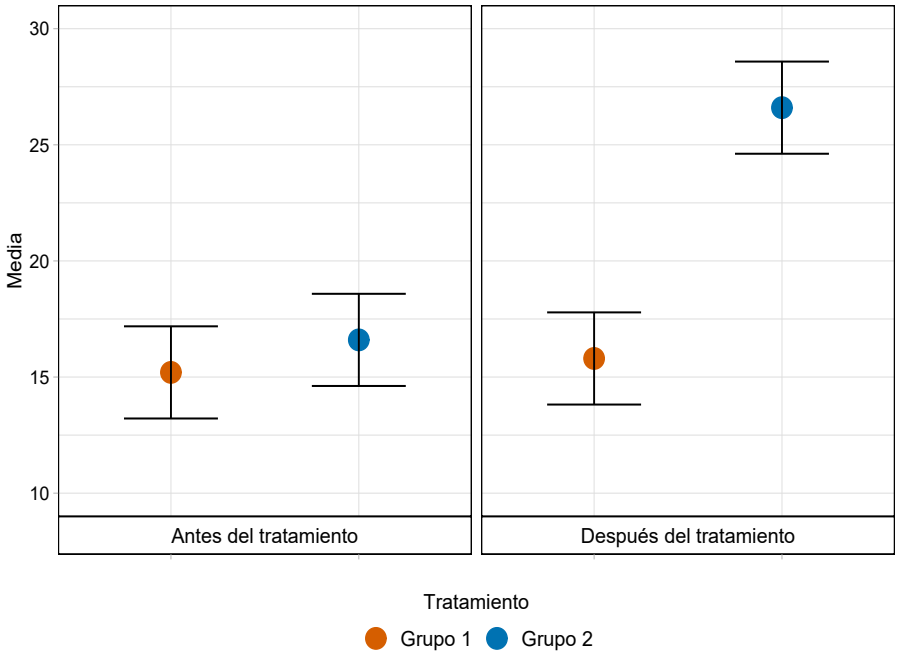
En otras ocasiones nos interesa obtener intervalos de confianza para los estimadores anteriores. En este caso podemos utilizar la función lincom() del paquete biostat3 para obtener intervalos de confianza para suma de estimadores.

```
library(biostat3)
# Grupo 2 post = 1
> lincom(did_reg, c("(Intercept) + Treatment + post + post:Treatment"))[1:3]
                                                 Estimate    2.5 %   97.5 %
(Intercept) + Treatment + post + post:Treatment     26.6 24.61569 28.58431
```

Y podemos construir una gráfica como en la Figura. 4[3]

[3]Para ver el código de la gráfica véase el Apéndice B.

Figura 1: Diagrama de Diferencias en Diferencias



Las imágenes creadas en ggplot se pueden guardar automáticamente.

```
# Guardamos imagen
ggsave(dd_plot, filename = 'imagen_dd.pdf', device = cairo_pdf, dpi = 300, width = 12,
     height = 10, units = 'cm')
```

5.2. Two-way fixed-effect regression

De acuerdo con Imai y Kim (2020), la regresión lineal bidireccional de efectos fijos (2FE) se ha convertido en un método predeterminado para estimar efectos causales a partir de datos panel. Muchos investigadores utilizan el modelo 2FE para ajustar doble efectos fijos: por unidad y por tiempo, que interactúan en un lapso determinado. La forma estándar del modelo es:

$$Y_{it} = \gamma_i + \lambda_t + \tau T_{it} + \epsilon_{it},$$

donde, γ_i es el efecto fijo por individuo, λ_t es el efecto fijo por periodo, T_{it} es una variable dicótoma igual a 1 si el individuo i es tratado para el tiempo $t \geq \sigma$, ϵ_{it} es el error aleatorio y τ es el parámetro de interés.

En R corremos el modelo 2FE de 2 formas distintas: utilizando la función **lm()** y **plm()** del paquete plm() publicado por en *Panel Data Econometrics in R: The plm Package*, Croissant y Millo (2008).

```
# Los datos se generan más adelante
# Forma 1 de obtener tau
forma_1 <- lm(y ~ factor(individual) + factor(year) + treatment, data = dd)

# Forma 2 de obtener tau
forma_2 <- plm(y ~ treatment , data = dd, model="within",effect = "twoways")
```

Lags and leads

Una estrategia para probar el supuesto de tendencias paralelas es correr una regresión de *Lags and leads*:

$$Y_{it} = \gamma_i + \lambda_t + \sum_{t' \in \mathbf{F}} \delta_{t'} T_{it+t'} + \epsilon_{it}, \quad \text{donde } \mathbf{F} = \{-f, ..., f\},$$

Donde γ_i y λ_t son los efectos fijos por individuo y periodo, respectivamente. $T_{it+t'}$ $\forall t' \in \mathbf{F}$ son las variables rezagadas y adelantadas, incluyendo el periodo 0, donde inicia

el tratamiento, y verificar que las variables adelantadas son estadísticamente igual a 0.

Una aplicación ilustrativa de Lags and leads se puede encontrar en Autor (2003) y una explicación más detallada del método se puede encontrar en las notas de clase de Pischke (2005).

Did plot

Una segunda alternativa para detectar tendencias antes del tratamiento y efectos anticipados es utilizando *Did plot*:

$$Y_{it} = \gamma_i + \lambda_t + \sum_{t'=-T}^{0} \delta_{t'} T_i + \sum_{t'=1}^{T} \delta_{-t'} T_i + \epsilon_{it}.$$

Para hacer el Did plot utilizamos el paquete did desarrollado por Brantly C. (1999).

Ejemplo 1: *No differential pre-trends*

Para este ejemplo, simulamos 20 países que recibieron una calificación anual del 2010 – 2019. El rango de la calificación va del 0, siendo al más baja, a l5 siendo la más alta. 10 de estos países implementaron una política en cualquier año desde el 2013 al 2017.

```
# Simulamos los datos para el grupo de tratamiento
tratados <- cbind.data.frame(c(rep(c(1:10), each=10)), (rep(c(2010:2019), 10)), rep
    (1,100), c(rep(c(2013,2014,2015,2016,2017),each = 20)))
names(tratados) <- c("state", "year", "treat", "year_treated")
set.seed(13) # Semilla aleatoria para replicacion

tratados <- tratados %>% group_by(state) %>% mutate(post_treatment = ifelse(row_number
    () >= which(year == year_treated), 1, 0))

# Generamos los datos del grupo de control
control <- cbind.data.frame(c(rep(c(11:20), each=10)), rep(c(2010:2019), 10))
names(control) <- c("state",  "year")

# Juntamos las 2 bases
dd <- bind_rows(tratados, control)

# Cambiamos los valores NA por 0
dd[is.na(dd)] <- 0

# Creamos multiples leads and lags
library(plm)
dd <- pdata.frame(dd,index = c("state","year")) # creamos formato pdata de plm, por
    facilidad
dd <- cbind(dd, lag(dd$post_treatment,-3:3)) # creamos lags and leads
dd <- as.data.frame(transform(dd)) # reveritmos el formato de pdata al formato
    original
names(dd)[(length(dd) - 6):length(dd)] <- c("lead_3", "lead_2", "lead_1", "treat_
    period", "lag_1", "lag_2", "lag_3") # cambiamos los nombres de las variables

# revertimos las variables que se hicieron factor con pdata, para uso del DID plot
library(varhandle)
dd <- unfactor(dd)
# Reemplazamos valores NA del grupo de control por 0.
dd[100:200, 6:12][is.na(dd[100:200, 6:12])] <- 0
```

```r
# Generamos los datos sin diferential pre-trend de la variable dependiente
set.seed(5) # Semilla aleatoria
pre_treat <- sample(c(0, 1, 2, 3, 4, 5), size = 100, replace = TRUE, prob = c
    (0.05,0.05,0.1,0.2,0.3,0.3))
pre_control <- pre_treat + 1
post <- sample(c(0, 1, 2, 3, 4, 5), size = 100, replace = TRUE, prob = c
    (0.3,0.3,0.2,0.1,0.05,0.05))

dd <- dd %>% group_by(state)  %>%
  mutate(rate_u = ifelse(state %in% c(1:10), create_y(post_treatment), pre_control)) #
      Posterior del tratamiento
```

Análisis utilizando *Lags and leads*:

```r
# Forma 1: utilizando lm
reg1 <- lm(rate_u ~ lead_3 + lead_2 + lead_1 + treat_period + lag_1 + lag_2 + lag_3 +
             factor(state) + factor(year), data = dd)

# Forma 2: utilizando el paquete plm y 2FE
reg2 <- plm(rate_u ~ lead_3 + lead_2 + lead_1 + treat_period + lag_1 + lag_2 + lag_3,
            data=dd, model="within", effect="twoways")
summary(reg2)

# Tabla comparativa
tabldedd <- stargazer(reg1, reg2,
                      header = FALSE,
                      font.size = "scriptsize",
                      dep.var.labels.include = FALSE,
                      table.placement = "H",
                      omit = c("Constant", "year", "state"),
                      column.labels = c("lm", "plm"),
                      covariate.labels = c("Lead 3", "Lead 2", "Lead 1", "Periodo 0",
                          "Lag 1", "Lag 2", "Lag 3"),
                      omit.stat = c("f", "ser","adj.rsq"),
                      title = "Leads and lags: comparacion de metodos",
                      type = "latex")
```

Tabla 4: *Leads and lags*: comparación de métodos

	lm	plm
	(1)	(2)
Lead 3	0.185	0.185
	(0.292)	(0.292)
Lead 2	−0.207	−0.207
	(0.227)	(0.227)
Lead 1	0.200	0.200
	(0.190)	(0.190)
Periodo 0	−1.973***	−1.973***
	(0.177)	(0.177)
Lag 1	0.501***	0.501***
	(0.190)	(0.190)
Lag 2	0.103	0.103
	(0.215)	(0.215)
Lag 3	1.236***	1.236***
	(0.308)	(0.308)
Observations	141	141
R^2	0.972	0.631
Note:	*p<0.1; **p<0.05; ***p<0.01	

Las variables en rojo son las variables de interés (las variables adelantadas) y observamos que no hay evidencia para rechazar la hipótesis nula $H0 : Lead_i = 0$, para $i = 1, 2, 3$.

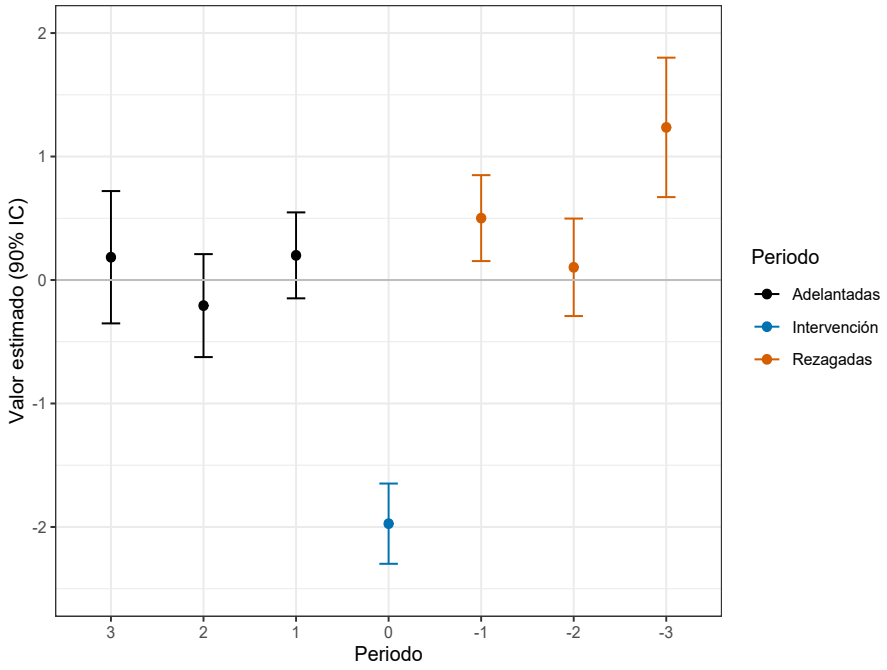
Gráficamente:

```
# Creamos datos para hacer la grafica de leads and lags
coeficientes <- coef(reg2)
se <- sqrt(diag((reg2$vcov)))
time <- 3:-3
# Concatenamos las variables
datos <- data.frame(coeficientes=coeficientes, ses = se, time = time, type = rep(1:3,
    c(3,1,3)))
```

```r
datos$time <- factor(datos$time, levels = c(3,2,1,0,-1,-2,-3))

# Gráfica de leads and lags
colors <- c("#000000", "#0072B2","#D55E00")
ggplot(data = datos, mapping = aes(y = coeficientes, x = time)) +
  geom_point(aes(colour = factor(type)), size = 2) +
  geom_errorbar(aes(ymin=(coeficientes-1.833*ses), ymax=(coeficientes+1.833*ses),
      colour = factor(type)), width=0.2) +
  ylim(c(-2.5,2)) +
  geom_hline(yintercept = 0, linetype="solid", color ="grey", 2) +
  geom_vline(xintercept = 0,linetype="dashed", color ="grey", 2) +
  theme_bw() +
  ylab("Valor estimado (90% IC)") +
  xlab("Periodo") +
  scale_x_discrete(labels = 3:-3, breaks = 3:-3) +
  scale_color_manual(name = "Periodo",  labels = c("Adelantadas","Intervención","
      Rezagadas"), values= colors)
```

Figura 2: Gráfica de *Leads and lags* con pre-tendencias paralelas



Análisis utilizando Did plot:

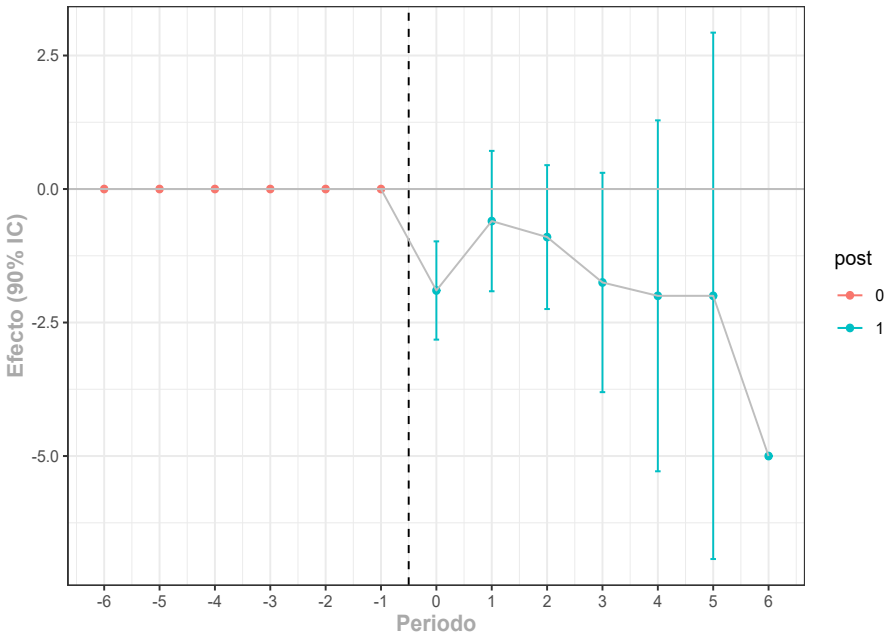
```
# Cargamos el paquete did
library(did)

# Utilizamos la funcion att_gt() para agregar los efectos
out <- att_gt(yname = "rate_u", gname = "year_treated", idname = "state", tname = "
    year", alp = 0.1, xformla = ~ 1, data = dd, est_method = "reg")

# Utilizamos la funcion aggte() para agregar los efectos entre periodos
es <- aggte(out, type = "dynamic")

# Utilizamos la funcion ggdid() del mismo paquete para graficar es
ggdid(es, title = " ") +
  geom_hline(yintercept = 0, linetype="solid", color ="grey", 2) +
  geom_vline(xintercept = -0.5,linetype="dashed", color ="black", 2) +
  xlab("Periodo") +
  ylab("Efecto (90% IC)") +
  geom_line(color="grey")
```

Figura 3: Gráfica DID con pre-tendencias paralelas



Note que no se viola el supuesto de tendencias paralelas, ya que las estimaciones para los periodos anteriores al tratamiento ($t < 0$) no son significativamente distintas de 0.

Ejemplo 1: *Differential pre-trends*

Usamos el mismo ejemplo anterior cambiando los valores de la variable dependiente para que el supuesto de pretendencias paralelas falle.

```
# Generar los datos sin diferential pre-trend
set.seed(6) # Semilla aleatoria

pre_treat <- sample(c(0, 1, 2, 3, 4, 5), size = 100, replace = TRUE, prob = c
    (0.05,0.05,0.1,0.2,0.3,0.3))
pre_control <- c(pre_treat[1:7], sample(c(0, 1, 2, 3, 4, 5), size = 3, replace = TRUE,
     prob = c(0.2,0.2,0.2,0.1,0.1,0.1)))
post <- sample(c(0, 1, 2, 3, 4, 5), size = 100, replace = TRUE, prob = c
    (0.3,0.3,0.2,0.1,0.05,0.05))

# Creamos la variable tasa de desempleo
dd <- dd %>% group_by(state)  %>%
  mutate(rate_u = ifelse(state %in% c(1:10),
                         c(pre_treat[1:(match(1,dd$post_treatment) + 1)], # Antes del
                             tratamiento
                           1, # Valor al momento de inciial el tratamiento
                           post[(match(1,dd$post_treatment) + 1):10]), pre_control)) #
                                Posterior del tratamiento
```

Análisis utilizando *Lags and leads*:

```
# Utilizando el paquete plm y 2FE
reg3 <- plm(rate_u ~ lead_3 + lead_2 + lead_1 + treat_period + lag_1 + lag_2 + lag_3,
            data=dd, model="within", effect="twoways")
summary(reg3)

# Tabla comparativa
tabldedd2 <- stargazer(reg3,
                       header = FALSE,
                       font.size = "scriptsize",
                       dep.var.labels.include = FALSE,
                       table.placement = "H",
                       omit = c("Constant", "year", "state", "lag_1", "lag_2", "lag_3"
                           ),
                       column.labels = c("Calificación"),
                       covariate.labels = c("Lead 3", "Lead 2", "Lead 1", "Periodo 0")
                           ,
                       omit.stat = c("f", "ser","adj.rsq"),
```

```
                        title = "Leads and lags: Differential pre-trends",
                        type = "latex")

note.latex <- "\\multicolumn{2}{l} {\\parbox[t]{6cm}{ \\textit{Notas:}
Las variables rezagadas se incluyen como controles, pero se omiten. * denota p$<$0.1,
    ** denota p$<$0.05, y *** denota p$<$0.01.}} \\\\"
tabldedd2[grepl("Note", tabldedd2)] <- note.latex

cat(tabldedd2)
```

Tabla 5: Leads and lags: Differential pre-trends

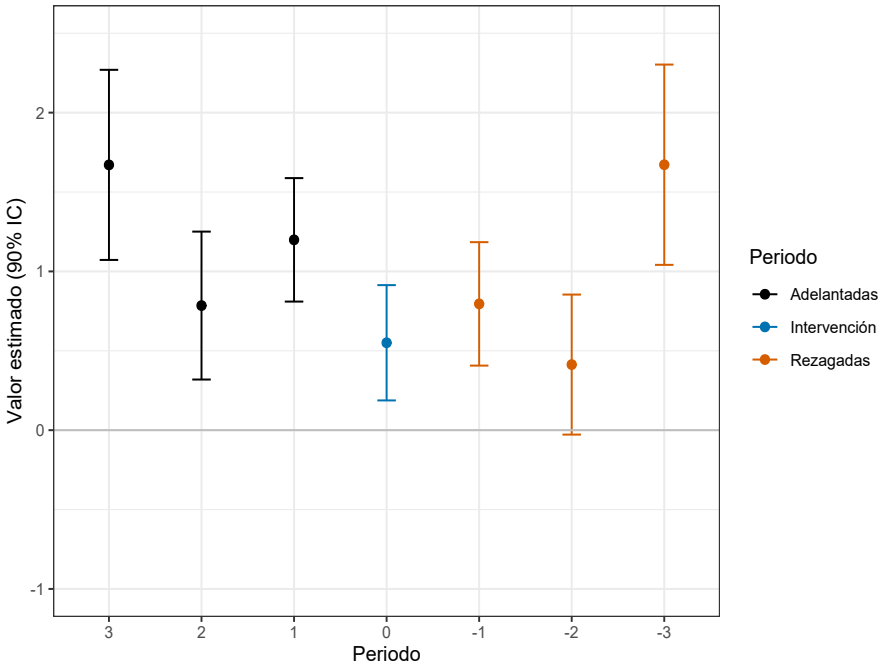
	Calificación
lead 3	1.671***
	(0.327)
lead 2	0.785***
	(0.254)
lead 1	1.199***
	(0.212)
Periodo 0	0.550***
	(0.198)
Observaciones	141
R^2	0.606

Notas: Las variables rezagadas se incluyen como controles, pero se omiten. * denota p<0.1, ** denota p<0.05, y *** denota p<0.01.

Observamos que las variables $Lead_1$, $Lead_2$ y $Lead_3$ son significativamente distintas de cero.

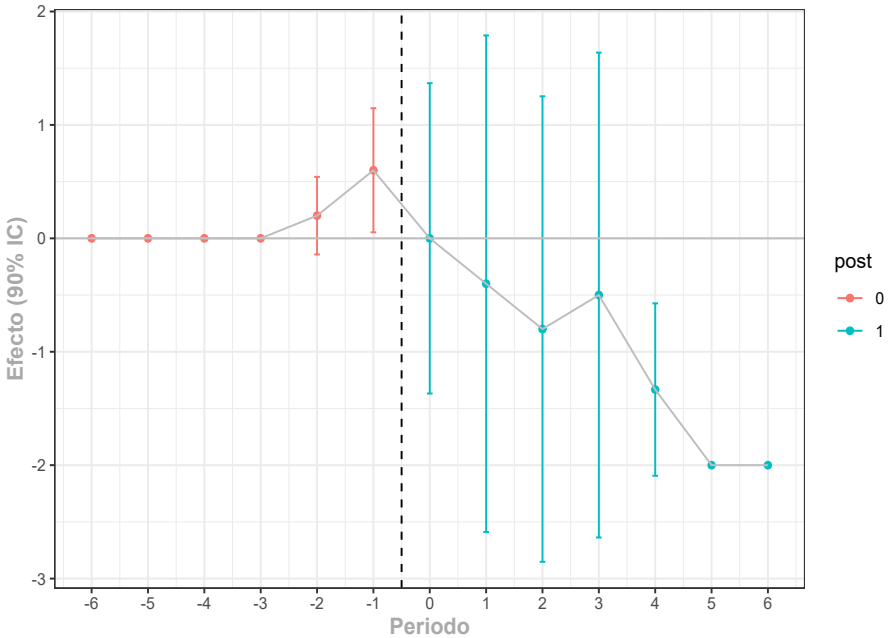
Gráficamente:

Figura 4: Gráfica de Leads and Lags sin pre-tendencias paralelas



Análisis utilizando Did plot:

Figura 5: Gráfica Did con pre-tendencias diferentes



Note que se viola el supuesto de tendencias paralelas, ya que las estimaciones para los periodos anteriores al tratamiento ($t < 0$) son significativamente distintas de 0.

6. Variables instrumentales

Conceptualmente, la estimación por variables instrumentales se puede interpretar como dos etapas independientes de mínimos cuadrados ordinarios:

1. Primer etapa

$$X \sim Z\delta + \epsilon,$$

   donde X son los predictores endógenos, Z los potenciales instrumentos y ϵ el vector de errores. Dado la estimación de $\hat{\delta}$, podemos obtener $\hat{X} = Z\hat{\delta}$.

2. Segunda etapa

$$Y \sim \hat{X}\beta_{IV} + \mu$$

   donde Y es la variable dependiente de interés y μ es el error.

Para efectos ilustrativos en R, generamos datos de 3 variables: salarios, educación y educación de los padres. Obsérvese que la educación de los padres es función de la variable educación.

```
# Creamos los datos aleatoriamente
set.seed(32)
salarios <- c(10, 10, 11, 13, 15, 18, 20, 20, 19, 21)
educacion <- c(1,1,1,2,1,2,3,3,3,3)
educacion_padres <- educacion - sample(c(0,1,2), size = 10, replace = T, prob = c
    (0.7,0.2,0.1))
datos_iv <- cbind.data.frame(salarios, educacion, educacion_padres)
```

Primero, hacemos mínimos cuadrados en dos etapas (MC2E). En el siguiente código calculamos primero el *First stage* tomando la variable educación como dependiente y educación de los padres como nuestro instrumento. Posteriormente, tomamos los valores ajustados de esta regresión y por último corremos la segunda etapa: la variable dependiente (salarios) sobre los valores ajustados obtenidos.

```r
# Paso a paso
fs <- lm(educacion ~ educacion_padres) # first stage

# Test weak instrument
linearHypothesis(fs, c("educacion_padres = 0"), test = "F")[6]$`Pr(>F)`[2]

# Otra manera de obtener el mismo resultado es mediante First Stage y Reduced form
fs <- lm(educacion ~ educacion_padres) # first stage
rf <- lm(salarios ~ educacion_padres) # reduced form

# Sacamos el cociente de las estimaciones
b_iv <- rf$coefficients[2]/fs$coefficients[2]
```

Note que esto es equivalente a regresar x sobre z (first stage), y sobre z (reduced form) y obtener β_{IV} como el cociente de las estimaciones, i.e.:

$$\beta_{IV} = \frac{\text{Est. Reduced form}}{\text{Est. First stage}}.$$

Note que en la práctica no podemos tomar los *predicted values* e incorporarlos en el *second stage* y por esto hacemos uso de paquetes que ya incluyen una correción a los errores estándar (porque ya incorpora el error en la estimación de la primera etapa) es usando el paquete AER o *Applied econometrics with R* y la función ivreg(). De igual forma tambien se puede hacer uso de la función plm y de la función **felm** del paquete **lfe**.

```r
ivreg(formula, instruments, data, subset, na.action, weights, offset,
  contrasts = NULL, model = TRUE, y = TRUE, x = FALSE, ...)
```

También podemos obtener pruebas estadísticas que nos dicen si estamos utilizando un buen instrumento utilizando la función **summary()** con el argumento *diagnostics* $= T$.

```r
# Hacemos lo mismo utilizando la función ivreg
library(AER)
reg_iv = ivreg(salarios ~ educacion| educacion_padres)

# Hacemos lo mismo utilizando la función ivreg y felm
reg_iv = AER::ivreg(salarios ~ educacion| educacion_padres, data =datos_iv)
reg_iv_felm = felm(salarios ~ 1 | 0 | (educacion ~ educacion_padres), data =datos_iv )
```

```
# obtenemos algunos test para ver si tenemos un buen instrumento
> summary(reg_iv, df = Inf, diagnostics = TRUE)

Call:
ivreg(formula = salarios ~ educacion | educacion_padres)

Residuals:
   Min     1Q Median     3Q    Max
-2.700 -1.436 -0.200  1.021  3.329

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   7.6429     1.9481   3.923 8.73e-05 ***
educacion     4.0286     0.9213   4.373 1.23e-05 ***

Diagnostic tests:
                 df1 df2 statistic p-value
Weak instruments   1   8    11.462 0.00956 **
Wu-Hausman         1   7     0.127 0.73232
Sargan             0  NA        NA      NA
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1         1

Residual standard error: 2 on Inf degrees of freedom
Multiple R-Squared: 0.8183, Adjusted R-squared: 0.7956
Wald test: 19.12 on 1 DF,  p-value: 1.227e-05
```

Tabla 6: Variables instrumentales

	OLS	*instrumental variable*	*felm*
	First Stage	Modelo ivreg	Modelo felm
	(1)	(2)	(3)
Educación estimada	0.673***		
	(0.199)		
Educación		4.029***	
		(0.921)	
Educación (felm)			4.029***
			(0.921)
Observations	10	10	10
R^2	0.589	0.818	0.818
Note:			*p<0.1; **p<0.05; ***p<0.01

Un análisis similar y más detallado de esta sección se puede encontrar en Colonescu (2018).

7. Regresión discontinua

Consideramos el modelo

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 Z_i + \delta X_i * Z_i + \mu_i,$$

donde Y_i denota la variable endógena, X_i una variable explicativa continua y Z_i una variable por partes, discontinua, y función de X_i.

$$Z_i = \begin{cases} 0 & X_i \leq \bar{x} \\ 1 & X_i > \bar{x}, \end{cases}$$

donde $\bar{x}$ es una constante que se conoce como el punto de quiebre. En este caso, todos los individuos tales que $X_i > \bar{x}$ reciben el tratamiento y aquellos que tienen un valor por debajo o igual al punto de quiebre, $X_i \leq \bar{x}$, no reciben el tratamiento. El estimador asociado a Z_i, β_2, es el *Average treatment effect* (ATE) de los individuos con $X_i = \bar{x}$.

En R haremos el análisis de dos formas: primero utilizando la función **lm()** como la hemos usado en secciones anteriores y posteriormente con el paquete rddtools, para hacer uso de la función rdd_data() y rrdd_reg_lm().

```
set.seed(4) # Semilla  aleatoria
# Generamos los datos
x <- runif(100, -2, 2)
y <- 1 +  x + 5*(x >= 0) + rnorm(100,0,1)
z <- ifelse(x > 0, 1, 0) # creamos la variable indicadora
# Concatenamos los vectores
rd_data <- cbind.data.frame(y,x,z)
```

McCrary (2008) desarrolló un *test* de densidad para probar la continuidad de la variable explicativa (*running variable*) en el punto de quiebre. Esta prueba verifica si la variable fue manipulada o no por los agentes en cuestión, es decir, verificamos si se cumple el supuesto de no manipulación. Una discontinuidad implica que los datos fueron manipulados. En términos básicos, la hipótesis nula es que la discontinuidad es 0.

Entonces, si el valor p es menor que .05, tenemos 95 % de confianza de que ha ocurrido una manipulación.

Una manera fácil de hacer la prueba de McCrary es con el paquete rdd

```
# Test de McCrary
library(rdd)
DCdensity(rd_data$x , 0, plot = F)
[1] 0.4241542
```

Corremos la regresión:

```
# Forma 1: utilizando la función lm():
rd1 <- lm(y ~ z + z*x , data = rd_data) # el punto de quiebre es 0
rd1$coefficients
(Intercept)           z           x         z:x
  0.9283781   4.8522043   0.9687941   0.1969636
```

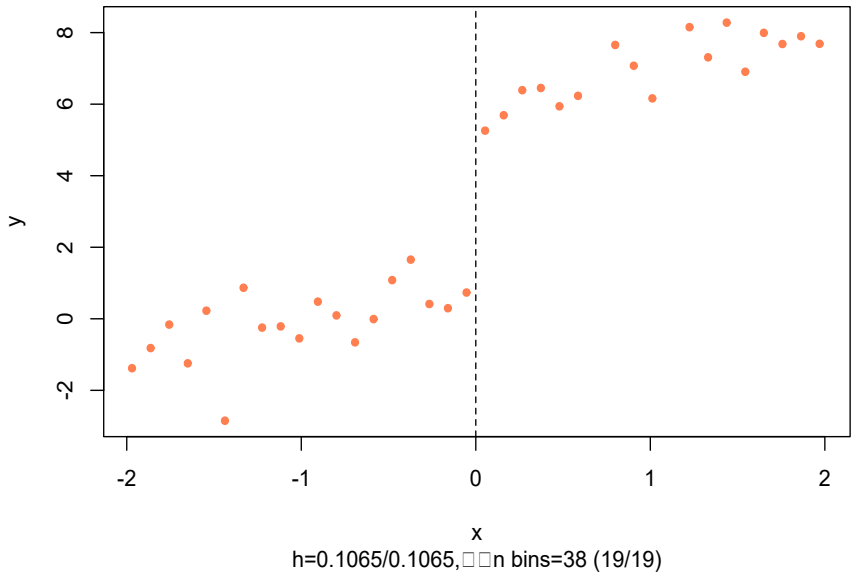
```
# Sin optimal bandwidth
rd2 <- rdd_reg_lm(rdd_object = rd_data, slope = "separate", bw = NULL)
rd2$coefficients
(Intercept)           D           x     x_right
  0.9283781   4.8522043   0.9687941   0.1969636
```

Uno de los beneficios de utilizar el paquete rddtools es que podemos ver rápidamente las gráficas de los datos creados y de los resultados obtenidos, véase 6 y 8[4].

```
# Graficamos los datos muestra
plot(rd_data, col = c("coral"), xlab = "x", ylab = "y")
```

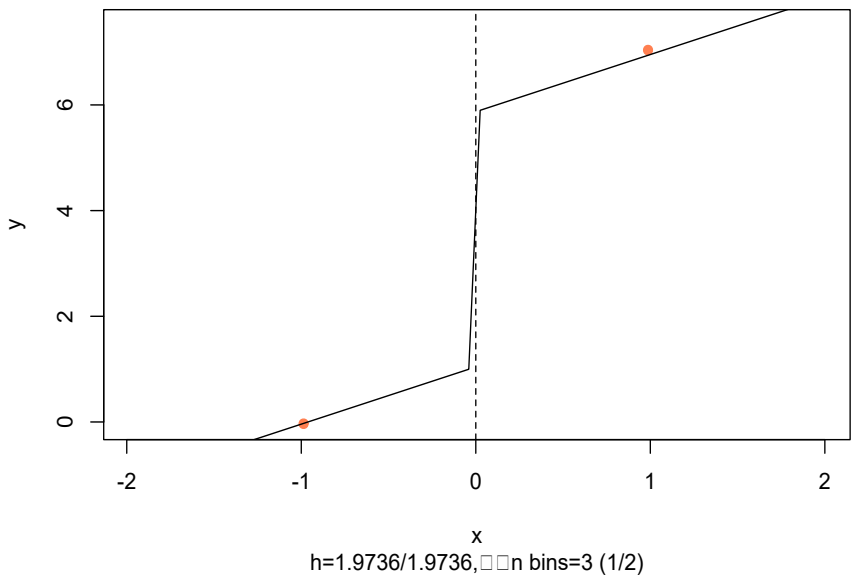
[4]En ambas gráficas los puntos representan promedios de observaciones agrupadas.

Figura 6: Datos muestra para el modelo de regresión discontinua




```
# Graficamos el modelo RD ajustado
plot(rd2, col = "coral",  xlab = "x", ylab = "y")
```

Figura 7: Modelo de regresión discontinua ajustado



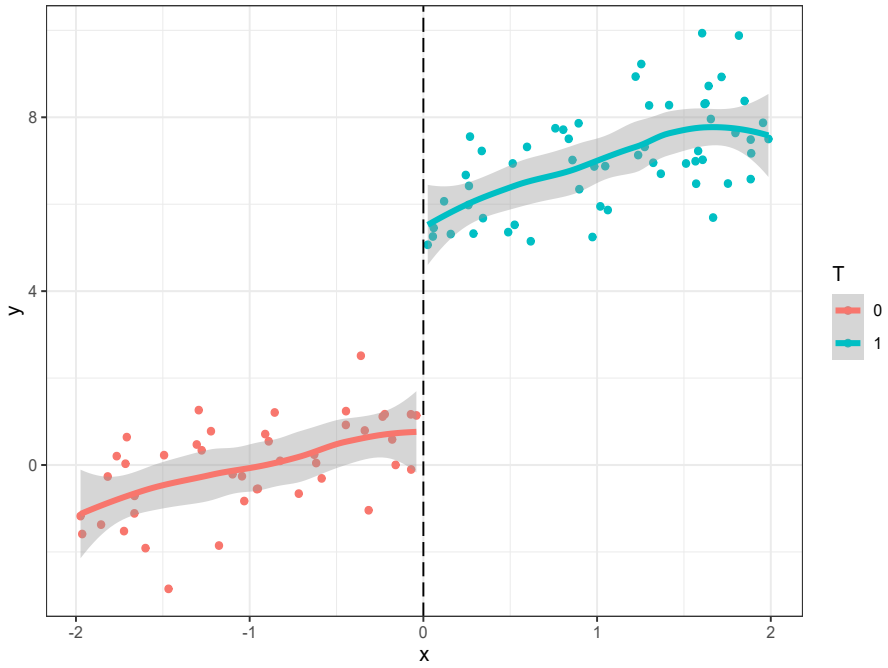
Observamos que obtenemos los mismos resultados al utilizar *lm*() y *rdd_reg_lm*(); sin embargo, no estamos considerando el parámetro de suavizamiento de un modelo de regresión discontinua. En Imbens y Kalyanaraman (2012) proponen una formula para el calculo del ancho de banda (*bandwidth*) óptimo. El calculo de este parámetro se implementa con la función rdd_bw_ik() del mismo paquete rddtools.

```
# RD Con optimal bandwidth
rd3_bw <- rdd_reg_lm(rdd_object = rd_data, slope = "separate", bw = rdd_bw_ik(rd_data,
     kernel = "Triangular"))
rd3_bw$coefficients
(Intercept)           D           x     x_right
  0.8891591   4.8656633   0.9065084   0.2972173
```

```
# Grafica RD simple
ggplot(rd_data, aes(x, y, color = factor(z))) +
  geom_point() +
  geom_smooth(size = 1.5) +
  geom_vline(xintercept=0, linetype="longdash") +
  theme_bw() +
  xlab("x") +
```

```
  ylab("y") +
  scale_color_discrete(name = "T")
```

Figura 8: Gráfica de regresión discontinua



Un análisis similar y más detallado de esta sección se puede encontrar en Hanck (2020).

Referencias

Autor, David H. (2003). «Outsourcing at Will: The Contribution of Unjust Dismissal Doctrine to the Growth of Employment Outsourcing». En: *Journal of Labor Economics* 21.

Brantly C., Pedro H. y Sant'Anna. (1999). URL: https://bcallaway11.github.io/did/ (visitado 16-08-2021).

Colonescu, Constantin (2018). *Principles of Econometrics with R*. Lulu.com. ISBN: 1387473611.

Croissant, Yves y Giovanni Millo (2008). «Panel Data Econometrics in R: The plm Package». En: *Journal of Statistical Software* 27.

Fundation, The R (2021). *What is R?* URL: https://www.r-project.org/about.html (visitado 15-08-2021).

Hanck Christoph, et al. (2020). *Introduction to econometrics with R*. Essen, Germany.

Imai, Kosuke e In Song Kim (2020). «On the Use of Two-Way Fixed Effects Regression Models for Causal Inference with Panel Data». En: *Cambridge University Press*.

Imbens, Guido y Karthik Kalyanaraman (2012). «Optimal Bandwidth Choice for the Regression Discontinuity Estimator». En: *The Review of Economic Studies* 21.

McCrary, Justin (2008). «Manipulation of the running variable in the regression discontinuity design: A density test». En: *Journal of Econometrics* 142(2), págs. 698-714.

Pischke, Steve (oct. de 2005). *Empirical Methods in Applied Economics*.

Smith, Olivia (2020). *Data Reshaping in R*. URL: https://www.datacamp.com/community/tutorials/data-reshaping-in-r (visitado 15-08-2021).

Team, R Development Core (2000). *Introducción a R*. 1.0.1.

8. Apéndice A

```
# Instalamos los paquete necesarios
list.of.packages <- c("stargazer", "xtable", "AER", "biostat3", "car", "ggplot2", "did
    ", "rddtools")

# Verificamos aquellos paquetes que no han sido instalados
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"
    Package"])]

# Los instalamos y cargamos todos a la vez
if(length(new.packages)) install.packages(new.packages) # verificamos que exista al
    menos un paquete en new.packages y lo instalamos
lapply(list.of.packages, library, character.only = TRUE) # Función en R que aplica la
    función library a todos los elementos en list.of.packages

 # Limpiamos el workspace
rm(list = ls())

# Establecemos el directorio de trabajo al directorio de este documento con el paquete
     rstudioapi
rstudioapi::getActiveDocumentContext
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
```

9. Apéndice B

```
# Obtenemos las estimaciones de las medias e intervalos de confianza al 5%.
# Grupo 1 post = 0
treat1_before <- lincom(did_reg, c("(Intercept)"))[1:3]
# Grupo 2, post = 0
treat2_before <- lincom(did_reg, c("(Intercept) + Treatment"))[1:3]
# Grupo 1 post = 1
treat1_after <- lincom(did_reg, c("(Intercept) + post"))[1:3]
# Grupo 2 post = 1
treat2_after <- lincom(did_reg, c("(Intercept) + Treatment + post + post:Treatment"))
    [1:3]

# Generamos un dataframe
data_est <- rbind.data.frame(treat1_before, treat2_before, treat1_after, treat2_after)

data_est <- cbind.data.frame(data_est, c(rep("Antes del tratamiento",2),
                                         rep("Después del tratamiento",2)),
                             c("Grupo 1", "Grupo 2", "Grupo 1", "Grupo 2"))

colnames(data_est) <- c("Estimate", "left", "right", "period", "Treatment")

library(ggplot2) # libreria necesaria

final.text.11 <- element_text(color = "black", size = 11, hjust = 0.5, family="Arial")
text.11 <- element_text(color = "black", size = 10)

# Definimos el tema: el fondo, color, tamaños de letra, etc.
th_ca <-  theme(strip.text.x = final.text.11,
                strip.placement = "inside",
                strip.background = element_rect(colour = "black", fill = "white"),
                axis.text.x = element_blank(),
                axis.text.y = text.11,
                axis.title.x=element_blank(),
                legend.text = final.text.11,
                legend.title = final.text.11,
                axis.text = final.text.11,
                panel.border = element_rect(colour = "black", fill = NA),
                legend.position="bottom")

colors_ca <- c("#D55E00","#0072B2")
positions <- c("Grupo 1", "Grupo 2")
```

```r
# Graficamos
dd_plot <- ggplot(data_est, aes(Treatment, Estimate), color = factor(Treatment)) +
  geom_point(aes(color = factor(Treatment)), size = 5, fill = "#d3d3d3") +
  geom_errorbar(aes(ymin = left, ymax = right), width = .5) +
  facet_grid(. ~ period, switch="both", labeller = labeller(variables = label_wrap_gen
      (25))) +
  scale_x_discrete(limits = positions) +
  theme_light() +
  th_ca +
  scale_color_manual(name = "Tratamiento",  labels = c("Grupo 1", "Grupo 2"),
                     values= colors_ca) +
  ylab("Media") +
  guides(color = guide_legend(title.position = "top", title.hjust = 0.5)) +
  geom_hline(yintercept=0, size = 0.8) +
  ylim(10, 30)
```