

# Notas de Econometría en R

Alvaro Carril\*

12 de junio de 2017

## Índice general

<b>1</b>	<b>Introducción</b>	<b>2</b>
1.1	Qué es R . . . . .	2
1.2	Obtener R . . . . .	2
1.3	Usar R interactivamente y a través de scripts . . . . .	3
<b>2</b>	<b>Trabajando con datos</b>	<b>3</b>
2.1	Objetos y asignaciones . . . . .	3
<b>3</b>	<b>Regresión lineal</b>	<b>3</b>
3.1	Mínimos Cuadrados Ordinarios . . . . .	5
3.2	Interpretando un modelo lineal . . . . .	8
3.3	Mínimos Cuadrados Ordinarios . . . . .	15
3.4	Extraer y calcular información de una regresión . . . . .	18
3.5	Heterocedasticidad . . . . .	20
3.5.1	Test de Breusch-Pagan . . . . .	20
3.5.2	Matriz de covarianza robusta . . . . .	21
3.6	Tests de hipótesis lineal . . . . .	22
3.7	Mínimos cuadrados ponderados . . . . .	23
3.8	Modelos con variables categóricas . . . . .	23
<b>4</b>	<b>Datos de panel</b>	<b>24</b>
4.1	Efectos fijos . . . . .	25

---

\*Estas notas son un intento de condensar lo justo y necesario para poder seguir un curso de econometría con R. Luego de recopilar material de muchas clases y ayudantías de econometría, en general pensadas para ser aplicadas en Stata, decidí que podía intentar hacer estas clases en R y usar esta guía como material de apoyo. Mi prioridad fue ser breve y conciso, por lo que estas notas no son un buen comienzo para aprender R (o econometría); son un complemento. Si hay algún error o quieres hacer algún comentario, mi correo es [acarril@fen.uchile.cl](mailto:acarril@fen.uchile.cl).

# 1 Introducción

## 1.1 Qué es R

R es un ambiente de software y un lenguaje de programación interpretado para hacer análisis estadístico y gráficos. Es una implementación de S, un lenguaje de programación matemático orientado a objetos más antiguo. Es software libre y de código abierto, activamente usado y ampliado por estadísticos y colaboradores de otras disciplinas.

R es mucho más flexible que la mayoría de los paquetes estadísticos normalmente usados por economistas. Es un lenguaje de programación completamente desarrollado, no sólo un programa con tests y métodos pre-programados.

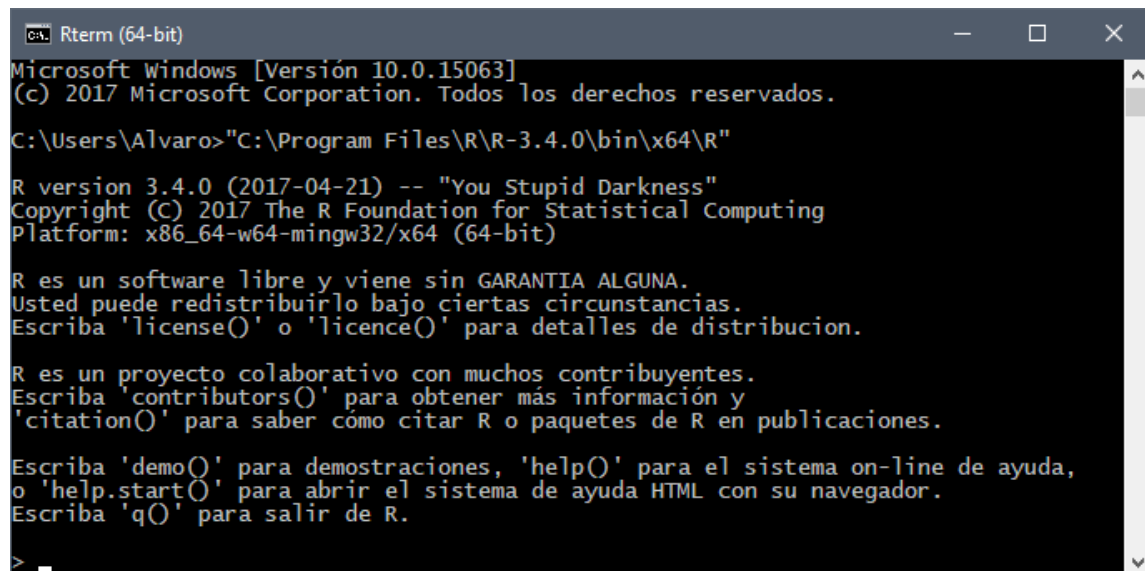
## 1.2 Obtener R

La instalación base de R puede obtenerse de <https://www.r-project.org/>. Una vez instalado puedes usar RGui para usarlo. Sin embargo, usualmente yo tomo dos pasos adicionales para obtener más flexibilidad. El primero es instalar RStudio, un IDE para R que incluye una consola, editor con resaltado de sintaxis, historial de comandos y varias otras vainas que lo hacen más útil que RGui, especialmente si eres principiante. Puedes descargar RStudio de <https://www.rstudio.com/>, y también es gratis.

Finalmente, a mi me gusta poder interactuar con R desde la línea de comandos, igual que puedo hacer con Python o Julia. Para poder hacer esto en Windows tenemos que ejecutar R:

```
"C:\Program Files\R\R-3.4.0\bin\x64\R"
```

Obviamente, tienes que reemplazar este directorio por el que corresponde a tu instalación.



```
C:\Users\Alvaro>"C:\Program Files\R\R-3.4.0\bin\x64\R"

R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

>
```

### 1.3 Usar R interactivamente y a través de scripts

```
getwd()
## [1] "/Users/alvaro/Dropbox (Personal)/Proyectos/NER"
```

## 2 Trabajando con datos

### 2.1 Objetos y asignaciones

R es un lenguaje interpretado, lo que significa que ejecuta nuestras instrucciones directamente, sin compilar un programa previo. Podemos usar R interactivamente a través de la consola:

```
1+2
## [1] 3
```

La mayoría de las operaciones y funciones en R no guardan el resultado de su ejecución. Por ejemplo, el resultado anterior (3) es calculado pero no puede ser reutilizado sin volverse a calcular. Para lograr esto tenemos que asignar el resultado de la operación a un objeto:

```
x <- 1+2
```

## 3 Regresión lineal

El modelo de regresión lineal es el caballo de batalla de la econometría aplicada. En su forma más simple usualmente se escribe como

$$y_i = \beta_0 + \beta_1 x_i + \mu_i. \quad (1)$$

donde  $i = 1, \dots, n$  es un índice que identifica a las observaciones (filas) en los datos. Podríamos tener  $n$  personas, familias o empresas. Los términos  $x$  e  $y$  son vectores de datos de tamaño  $n$ .

En este modelo suponemos que la variable dependiente  $y$  es una función lineal de la variable independiente,  $x$ . Entonces intentaremos buscar una combinación de parámetros  $\beta_0$  y  $\beta_1$  que se ajuste a nuestros datos. Como es posible que los datos no tengan una relación lineal perfecta, el modelo incluye un término de error  $\mu_i$ . Este vector indica la discrepancia entre la predicción del modelo y el valor observado, para cada unidad  $i$ .

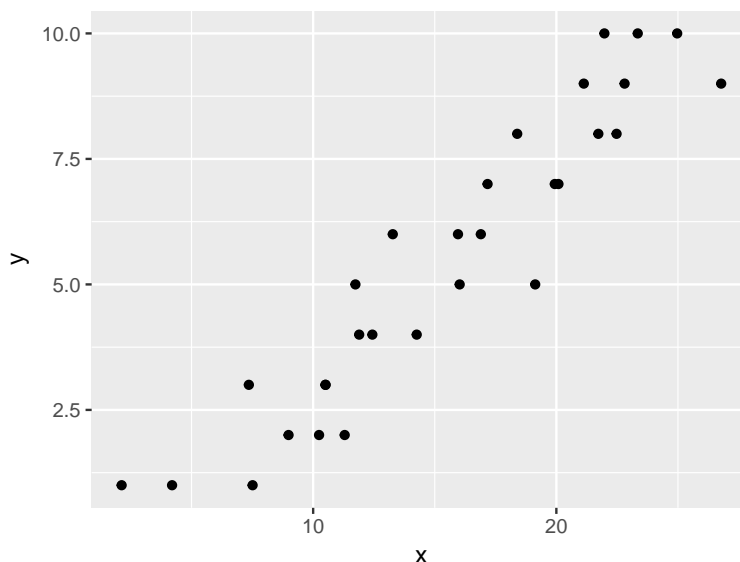
Por ejemplo, podríamos tener un vector  $y$  con los salarios de  $n$  personas y modelarlo como una función lineal de los años de educación de esas personas,  $x$ . Resulta útil tener una base de datos concreta con estos vectores, por lo que usaremos

```
simdatos <- read.csv(
  "https://raw.githubusercontent.com/acarril/NER/master/datos/simdatos.csv")
simdatos
```

El comando `read.csv()` permite leer bases de datos en formato CSV. Lo usamos para leer una base en línea y luego asignamos esos datos al objeto `simdatos` (datos simulados). Al escribir `simdatos` imprimimos los datos, y vemos que tienen 30 observaciones ( $n = 30$ ; omití el resultado aquí). La variable `y` representa ingreso por hora (en miles de pesos) y la variable `x` representa años de educación.

Podemos graficar estos datos usando **ggplot2**, un paquete muy poderoso para crear gráficos. Lo cargamos y luego creamos un gráfico de puntos con los datos:

```
library(ggplot2)
ggplot(simdatos, aes(x, y)) + geom_point()
```

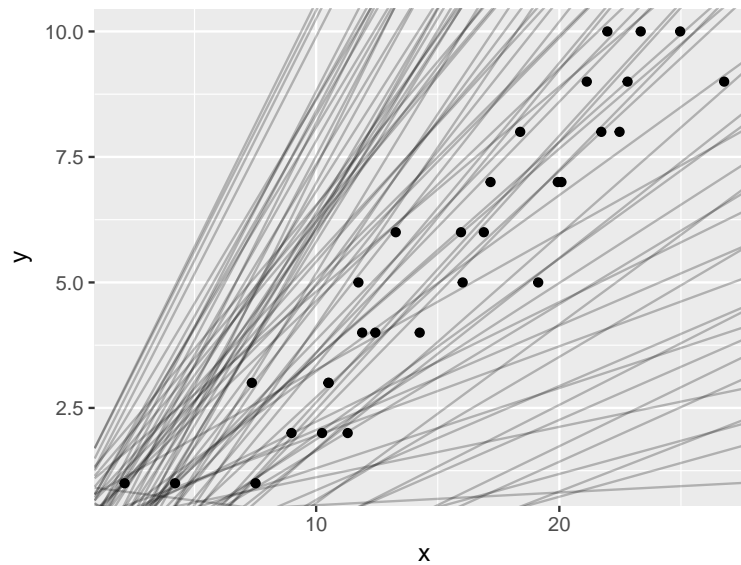


Vemos que existe una evidente relación lineal entre  $x$  e  $y$ : a mayor nivel de educación parece haber mayor nivel de ingreso. Cabe preguntarse ahora cuál es la manera óptima de elegir los parámetros que determinan dicha relación lineal, es decir, ¿cómo podemos elegir  $\beta_0$  (el intercepto) y  $\beta_1$  (la pendiente) en (1) para que la línea de ajuste sea la mejor posible?

Podríamos elegir entre una infinita variedad de combinaciones de  $\beta_0$  y  $\beta_1$  para ajustarse a los datos. Por ejemplo, el código de abajo simula 150 líneas con interceptos y pendientes “razonables” para estos datos:

```
set.seed(314)
modelos <- data.frame(
  beta1 = runif(150, -3, 1),
  beta2 = runif(150, -1, 1)
)
ggplot(simdatos, aes(x, y)) +
  geom_point() +
  geom_abline(
    aes(intercept = beta1, slope = beta2),
```

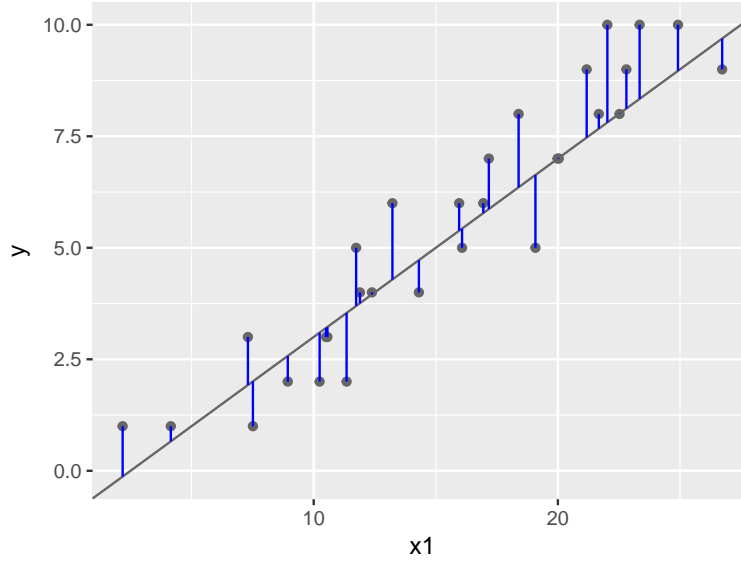
```
data = modelos, alpha = 1/4  
)
```



Varias de estas líneas parecen ajustarse razonablemente bien a los datos, por lo que necesitamos un criterio riguroso para elegir los parámetros  $\beta_0$  y  $\beta_1$  que definen estas líneas. Esto es lo que haremos a continuación.

### 3.1 Mínimos Cuadrados Ordinarios

El método de Mínimos Cuadrados Ordinarios (MCO) es una respuesta directa a la pregunta que planteamos recién: ¿qué criterio conviene usar para elegir el modelo lineal que se ajuste “mejor” a los datos? Lo que propone MCO es minimizar la suma de los errores al cuadrado. Geométricamente, esto equivale a minimizar la distancia vertical entre los puntos y la recta definida por el modelo.



La distancia entre un puntos y la recta el error de predicción del modelo, capturado por  $\mu_i$ . Es decir que para cada observación  $i$  tenemos una medida de la magnitud del error que esta elección de  $\beta_0$  y  $\beta_1$  produce. El criterio de MCO entonces es elegir  $\beta_0$  y  $\beta_1$  tal que se minimice el promedio de los errores al cuadrado.<sup>1</sup> Entonces el problema de minimización de MCO puede plantearse como

$$\begin{aligned} \min_{\beta_0, \beta_1} \sum_{i=1}^n \mu_i^2 \\ \Leftrightarrow \min_{\beta_0, \beta_1} \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2. \end{aligned} \quad (2)$$

Al resolver este problema obtenemos los estimadores MCO:

$$\begin{aligned} \hat{\beta}_1 &= \sum_{i=1}^n \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \\ &= \frac{\text{Cov}(x, y)}{\text{Var}(x)} \end{aligned} \quad (3)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (4)$$

donde  $\bar{x}$  e  $\bar{y}$  son los promedios de  $x_i$  e  $y_i$ ,  $\text{Var}(x)$  es la varianza muestral de  $x$  y  $\text{Cov}(x, y)$  es la covarianza muestral de  $x$  e  $y$ .

---

<sup>1</sup>¿Por qué al cuadrado? Porque se asume que es irrelevante si el error es una subestimación o una sobreestimación, de forma que lo único que importa es la magnitud. ¿Por qué no se usa el valor absoluto entonces? Buena pregunta: <https://stats.stackexchange.com/q/46019/91358>.

Esto significa que para encontrar la mejor línea de ajuste simplemente tenemos que calcular  $\bar{y}$ ,  $\bar{x}$ ,  $\text{Cov}(x,y)$  y  $\text{Var}(x,y)$ . ¡Hagámoslo ahora!

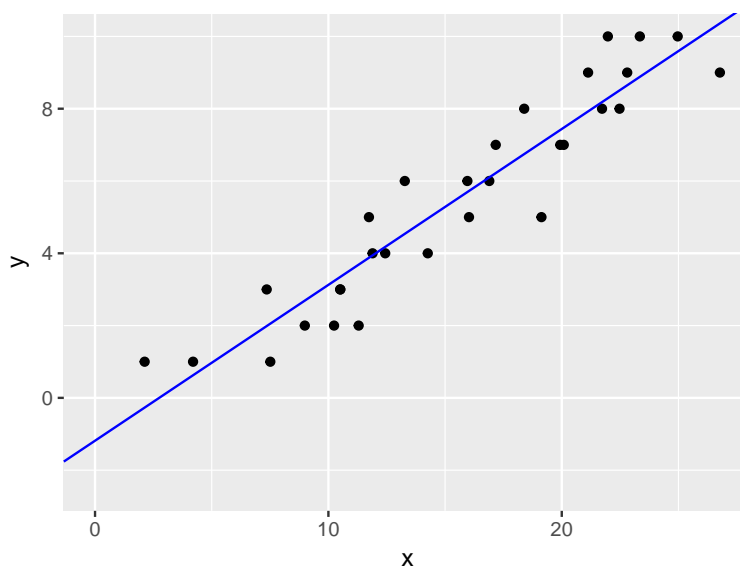
```
# Indicar base de datos a usar
attach(simdatos)
# Promedios de 'x' e 'y'
x.barra <- mean(x)
y.barra <- mean(y)
# Varianza de 'x' y covarianza entre 'x' e 'y'
var.x <- var(x)
cov.xy <- cov(x,y)
# Parámetros MCO:
(beta1 <- cov.xy/var.x)

## [1] 0.4311958

(beta0 <- y.barra - beta1*x.barra)

## [1] -1.185369

# 'Desactivar' base de datos
detach(simdatos)
# Graficar línea de regresión MCO
ggplot(simdatos, aes(x,y)) +
  expand_limits(x = 0, y = -2.5) +
  geom_point() +
  geom_abline(
    intercept = beta0, slope = beta1,
    color = "blue", show.legend = FALSE )
```



Vemos que es fácil calcular los estimadores MCO y en este caso  $\hat{\beta}_0 = 0.43$  y  $\hat{\beta}_1 = -1.19$ . La línea roja corresponde a la regresión MCO,

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x.$$

Si bien el método que acabamos de usar es útil para entender de manera más profunda qué es lo que estamos haciendo al estimar un modelo lineal por el método de MCO, no es demasiado sucinto. Podemos obtener los parámetros directamente usando la función `lm()` de la siguiente forma:

```
lm(y ~ x, data = simdatos)

##
## Call:
## lm(formula = y ~ x, data = simdatos)
##
## Coefficients:
## (Intercept)          x
##      -1.1854       0.4312
```

La función `lm()` viene de *linear model*, y permite calcular rápidamente los parámetros de un modelo lineal. El primer argumento de la función es una fórmula de R, que no es exactamente lo mismo que una fórmula en el sentido usual de la palabra. Por el momento basta entender que a la izquierda de `~` indicamos la variable dependiente, mientras que a la derecha indicamos las variables independientes. Entonces `lm()` toma una fórmula como `y~x` y la traduce automáticamente a algo como `y = beta0 + beta1 * x`.

Una de las ventajas de usar `lm()` es que podemos guardar sus resultados en un objeto, el que luego podemos manipular para extraer otro tipo de información útil del modelo. A continuación veremos cómo extraer, manipular e interpretar esta información, lo que nos dará una comprensión mucho más profunda de lo que estamos haciendo.

## 3.2 Interpretando un modelo lineal

Usaremos ahora la base `wage2`, que es una de las bases usadas por Wooldridge (2013) y contiene información de ingresos (y otras variables) para 935 personas. Esta base es un subconjunto de los datos usados en Blackburn y Neumark (1992).

```
library(foreign)
ingresos <- read.dta("http://fmwww.bc.edu/ec-p/data/wooldridge/wage2.dta")
ingresos <- ingresos[c("wage", "educ")]
```

El paquete **foreign** nos permite leer bases de datos en formatos de otros programas. En este caso usamos la función `foreign::read.dta()` para leer una base de datos de Stata. Por el momento nos interesan dos variables: `wage`, que es un vector de ingresos (en dólares mensuales) y `educ`, que es un vector con años de escolaridad de cada persona. La segunda línea guarda solamente esas dos variables en el objeto `ingresos`.

Al cargar nuevos datos yo siempre recomiendo:



1. Imprimir el encabezado
2. Imprimir un resumen
3. Graficarlos

Escribimos lo siguiente para lograr estas tres cosas:

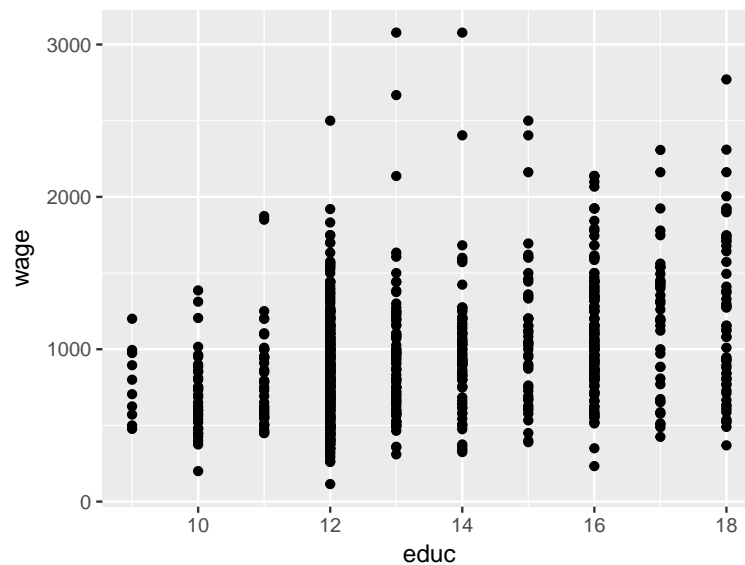
```
head(ingresos)

##   wage educ
## 1  769   12
## 2  808   18
## 3  825   14
## 4  650   12
## 5  562   11
## 6 1400   16

summary(ingresos)

##           wage           educ
##  Min.      : 115.0   Min.    : 9.00
## 1st Qu.: 669.0   1st Qu.:12.00
##  Median : 905.0   Median :12.00
##   Mean   : 957.9   Mean    :13.47
## 3rd Qu.:1160.0   3rd Qu.:16.00
##   Max.   :3078.0   Max.    :18.00

ggplot(ingresos, aes(educ, wage)) + geom_point()
```



Es evidente que estos datos (reales) no presentan una relación lineal tan evidente. Más adelante nos haremos cargo de este problema. Sin embargo, por ahora podemos usar `lm()` como ya lo

hemos hecho, definiendo una fórmula donde **wage** es la variable dependiente y **educ** es la variable independiente. Encerramos `lm()` dentro de `summary()` para obtener información más detallada de las estimaciones del modelo.

```
summary(lm(wage ~ educ, data = ingresos))

##
## Call:
## lm(formula = wage ~ educ, data = ingresos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -877.38 -268.63  -38.38  207.05 2148.26
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  146.952      77.715   1.891   0.0589 .
## educ         60.214       5.695  10.573   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 382.3 on 933 degrees of freedom
## Multiple R-squared:  0.107, Adjusted R-squared:  0.106
## F-statistic: 111.8 on 1 and 933 DF,  p-value: < 2.2e-16
```

Los coeficientes estimados de un modelo lineal como este pueden interpretarse fácilmente, recordando que nuestra predicción es

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x.$$

El intercepto de un modelo lineal nos indica que si las variables independientes son todas iguales a 0, el modelo predice, en promedio, un valor de  $\hat{\beta}_0$  para  $y$ . En términos de este modelo en particular, la interpretación es que si una persona tiene 0 años de educación el modelo predice un ingreso mensual promedio de 146.95 dólares.

Los coeficientes estimados para las variables independientes ( $\hat{\beta}_1$ , en este caso) pueden interpretarse como el efecto promedio que un aumento de una unidad de  $x$  tiene sobre  $y$ . Por ejemplo, nuestra estimación indica que un año adicional de educación tiene un efecto promedio de aumentar en 60.21 dólares el ingreso de las personas. Si este coeficiente fuera negativo diríamos que un cambio en  $x$  está correlacionado a una reducción de  $y$ .

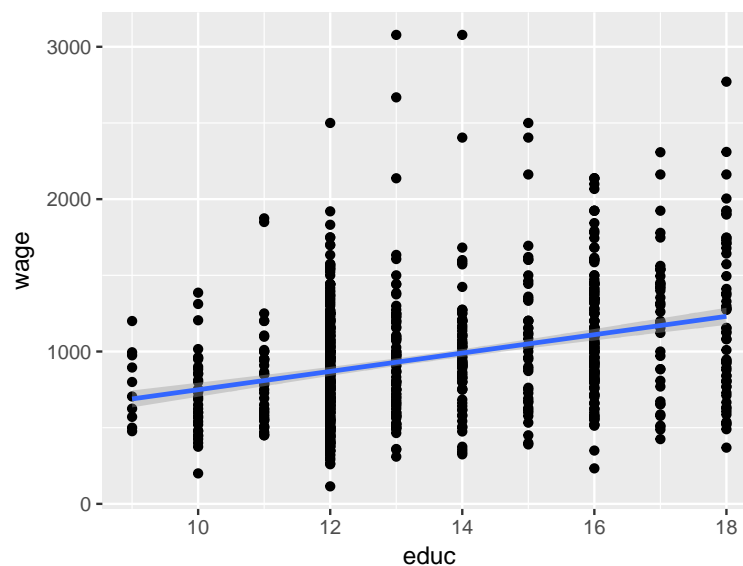
Los parámetros estimados, tal como los coeficientes de MCO, son estadísticos muestrales que usamos para hacer inferencias de los parámetros poblacionales. Es importante tener en cuenta que son estos parámetros poblacionales los que nos interesan en realidad, pero como no podemos observarlos directamente, debemos inferirlos por medio de una muestra finita.

Es evidente que si tomáramos una muestra distinta a la actual, lo más probable es que estimáramos parámetros diferentes. Si tomaras datos de otras 935 personas (o cualquier número, en realidad) es muy posible que al aplicar el mismo modelo obtengas valores de  $\beta_0$  y  $\beta_1$  distintos de 146.95 y 60.21. Además es muy probable que nuevamente ninguno de los dos sea igual al verdadero parámetro

poblacional, que es el que nos interesa. Si continuáramos este proceso de tomar diferentes muestras y estimar los mismos parámetros una y otra vez, veríamos que la frecuencia relativa de las estimaciones obtenidas sigue una distribución de probabilidad, y por el Teorema Central del Límite sabemos que es probable que esta distribución sea normal. Entonces es importante para nuestra estimación que cuantifiquemos la cantidad de incertidumbre en esta distribución poblacional desconocida. Ahí es donde entra el concepto de error estándar, que corresponde a nuestra estimación de la desviación estándar de la distribución de estos muestreos. En términos intuitivos, es una medida de la incertidumbre de  $\hat{\beta}$ .

En el ejemplo,

```
ggplot(ingresos, aes(educ, wage)) +
  geom_point() +
  geom_smooth(method = "lm")
```



Para guardar el modelo asignamos `lm()` a un objeto:

El objeto `modelo` no solo incluye los coeficientes estimados, si no que una serie de otros elementos que son muy útiles —veremos ahora por qué. En estricto rigor `modelo` es lo que R llama una lista, y contiene una serie de otros objetos. Podemos imprimir los nombres de estos objetos escribiendo

```
names(modelo)
## [1] "coefficients" "residuals"    "effects"      "rank"
## [5] "fitted.values" "assign"       "qr"          "df.residual"
## [9] "xlevels"      "call"        "terms"       "model"
```

Para acceder a los objetos guardados usamos la notación de `$`, como es usual en R. Por ejemplo, para imprimir los coeficientes del modelo escribimos

```
modelo$coefficients
## (Intercept)      educ
##          147         60
```

Además de estos elementos directamente accesibles guardados en `modelo`, existen una serie de funciones que podemos aplicar al objeto para realizar otros cálculos o resumir información útil. Por ejemplo, es muy común usar `summary()` para obtener un resumen de información importante del modelo estimado:

```
summary(modelo)

##
## Call:
## lm(formula = wage ~ educ, data = ingresos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -877.4 -268.6  -38.4   207.0 2148.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   146.95      77.71     1.89   0.059 .
## educ          60.21       5.69    10.57  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 382 on 933 degrees of freedom
## Multiple R-squared:  0.107, Adjusted R-squared:  0.106
## F-statistic: 112 on 1 and 933 DF, p-value: <2e-16
```

---

El modelo de regresión lineal es el caballo de batalla de la econometría aplicada, y puede escribirse como

$$y_i = x_i^T \beta + \epsilon_i, \quad (5)$$

donde  $i = 1, \dots, n$  es un índice que identifica a las observaciones (filas) en los datos. Podríamos tener  $n$  personas, familias o empresas. También es común escribir el mismo modelo en forma matricial:

$$y = X\beta + \epsilon. \quad (6)$$

En estas ecuaciones  $y$  (o  $y_i$ ) es un vector  $n \times 1$  que representa la variable dependiente del modelo, es decir, la variable que queremos explicar. Por ejemplo,  $y$  podría ser un vector de salario cuyos valores son los ingresos mensuales de  $n$  personas. La variable dependiente es explicada por un conjunto de  $k$  variables independientes, usualmente denotadas por  $X$ . Entonces  $X$  es una matriz  $n \times k$ , ya que contiene información de  $k$  variables (columnas) para  $n$  observaciones (filas).

Al estimar este tipo de modelos usualmente nos interesa determinar  $\beta$ , que representa la relación lineal entre  $y$  y  $X$ , es decir, entre la variable dependiente y el conjunto de variables independientes

que la expliquen. El vector  $\beta$  contiene  $k \times 1$  coeficientes de regresión, de forma que estimaremos un coeficiente por cada variable incluida en  $X$ . Finalmente  $\epsilon$  es el llamado término de error, un vector  $n \times 1$  que captura las discrepancias para cada observación  $i$  que resulta de imponer una relación lineal entre  $X$  e  $y$ .

Para entender bien todo esto resulta muy útil manipular un modelo simple. Para esto usaremos **sim1**, una base de datos simulada que contiene dos variables: **x** e **y**. Cargamos el paquete **modelr** para poder acceder esta base de datos, y además cargamos **ggplot2** para crear gráficos:

```
library(modelr)
library(ggplot2)
```

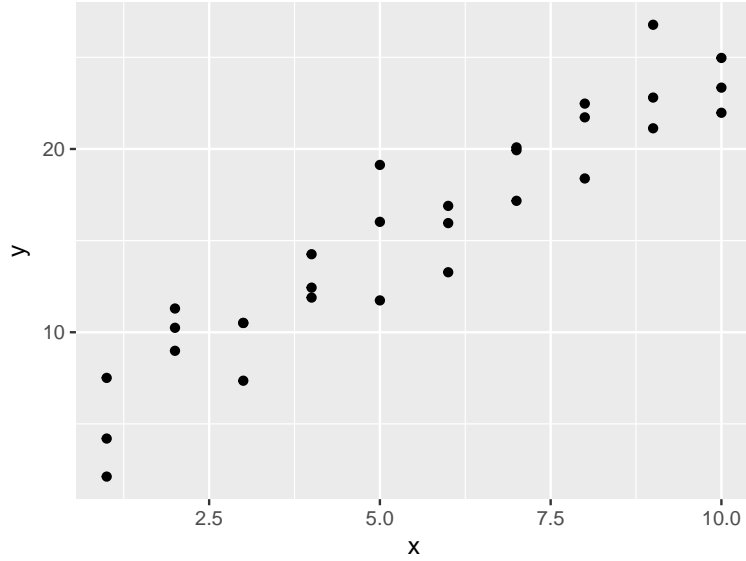
Primero que nada conviene imprimir el encabezado de la base de datos, que corresponde a las primeras observaciones.

```
sim1
## # A tibble: 30 x 2
##       x     y
##   <int> <dbl>
## 1     1  4.2
## 2     1  7.5
## 3     1  2.1
## 4     2  9.0
## 5     2 10.2
## 6     2 11.3
## 7     3  7.4
## 8     3 10.5
## 9     3 10.5
## 10    4 12.4
## # ... with 20 more rows
```

Esto es lo primero que debieses hacer antes de empezar a utilizar datos nuevos, ya que permite tener una idea rápida sobre ellos. Vemos que hay 30 observaciones ( $n = 30$ ) y que **x** solo toma valores enteros (y hay valores repetidos), mientras que **y** es continua.

Ahora graficamos estos datos para hacernos una idea de la relación que existe entre ambos:

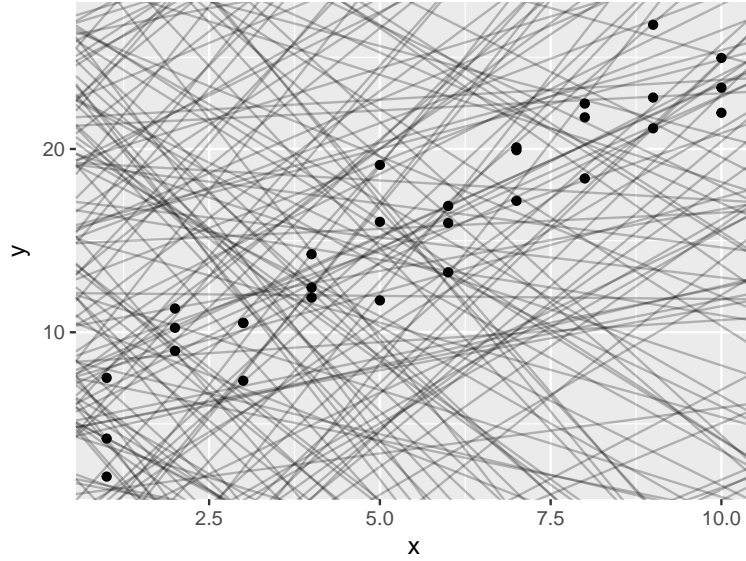
```
ggplot(sim1, aes(x, y)) + geom_point()
```



Oh sorpresa, los datos simulados presentan una clara correlación positiva. Parece casi diseñado para ser modelado linealmente... Recordemos que en este caso nuestro modelo lineal puede escribirse simplemente como

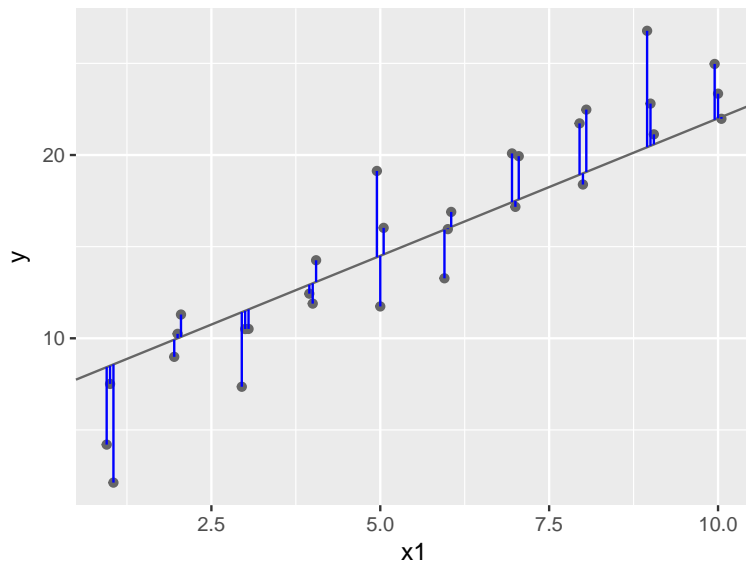
$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Entonces la pregunta relevante es “¿qué línea es la que mejor captura la relación entre  $x$  e  $y$ ?”. Notemos que en este caso bidimensional simple  $\beta_0$  y  $\beta_1$  representan el intercepto y la pendiente de la recta, respectivamente. Por lo tanto la pregunta también puede plantearse como: ¿cuáles son los valores de  $\beta_0$  y  $\beta_1$  que describen mejor esta relación?



### 3.3 Mínimos Cuadrados Ordinarios

El método de Mínimos Cuadrados Ordinarios (MCO) es una respuesta directa a la pregunta que planteamos recién: ¿qué criterio conviene usar para elegir el modelo lineal que se ajuste “mejor” a los datos? Lo que propone MCO es minimizar la suma de los errores al cuadrado. Geométricamente, esto equivale a minimizar la distancia vertical entre los puntos y la recta definida por el modelo.



La distancia entre un puntos y la recta el error de predicción del modelo, capturado por  $\epsilon_i$ . Es decir

que para cada observación  $i$  tenemos una medida de la magnitud del error que esta elección de  $\beta_0$  y  $\beta_1$  produce. El criterio de MCO entonces es elegir  $\beta_0$  y  $\beta_1$  tal que se minimice el promedio de todos los errores al cuadrado.<sup>2</sup>

---

```
library(AER)
data("CPS1988")
```

Consideremos la vieja y confiable ecuación de salarios de Mincer:

$$\text{salario}_i = \beta_0 + \beta_1 \text{educación}_i + \beta_2 \text{experiencia}_i + \epsilon_i \quad (7)$$

En términos de la base CPS1988, nos interesan por el momento las variables `wage`, `education` y `experience`. Podemos estimar este modelo lineal simple escribiendo

```
lm(wage ~ education + experience, data = CPS1988)

##
## Call:
## lm(formula = wage ~ education + experience, data = CPS1988)
##
## Coefficients:
## (Intercept)    education    experience
##      -385.1         60.9         10.6
```

El resultado que imprime `lm()` es una versión muy abreviada de todo lo que fue calculado, ya que sólo entrega las estimaciones de los coeficientes  $\beta$  del modelo. Para ver más detalles podemos guardar los resultados de `lm()` en el objeto `mincer` y luego aplicarle la función `summary()`:

```
mincer <- lm(wage ~ education + experience, data = CPS1988)
summary(mincer)

##
## Call:
## lm(formula = wage ~ education + experience, data = CPS1988)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1136    -221     -48     154   18156
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -385.083    13.243   -29.1   <2e-16 ***
## education    60.896     0.883    69.0   <2e-16 ***
## experience   10.606     0.196    54.2   <2e-16 ***
## ---
```

---

<sup>2</sup>¿Por qué al cuadrado? Porque se asume que es irrelevante si el error es una subestimación o una sobreestimación, de forma que lo único que importa es la magnitud. ¿Por qué no se usa el valor absoluto entonces? Buena pregunta.



```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 412 on 28152 degrees of freedom
## Multiple R-squared:  0.177, Adjusted R-squared:  0.177
## F-statistic: 3.02e+03 on 2 and 28152 DF,  p-value: <2e-16
```

Normalmente se usa la función `summary()` en objetos de datos, tales como vectores o marcos de datos, para obtener estadística descriptiva del objeto (promedio, desviación estándar, etc.). Al usarla en un objeto con resultados de una estimación —como en este caso— nos entregará información importantes de dicha estimación, como el  $R^2$  del modelo o los errores estándar de los coeficientes.

Podemos extraer otro tipo de información y objetos del modelo. La función `residuals()` entrega los residuos del modelo, y `vcov()` entrega la matriz de varianzas-covarianzas de los coeficientes:

```
# Residuos
residuals(mincer)
# Matriz de varianzas-covarianzas:
vcov(mincer)
```

Obviamente estas funciones sólo sirven para realizar otras computaciones y no para ser impresas directamente, y es por eso que he omitido su resultado.

Es común querer estimar un modelo sin intercepto, lo que se logra escribiendo el intercepto explícitamente en la fórmula, usualmente como 0:

```
lm(wage ~ 0 + education + experience, data = CPS1988)

##
## Call:
## lm(formula = wage ~ 0 + education + experience, data = CPS1988)
##
## Coefficients:
##  education  experience
##      36.55       7.65
```

Podemos especificar formas funcionales más complejas para nuestro modelo básico expresado en (7), incluyendo términos interactuados con `*` y términos cuadráticos encerrados en la función `I()`:

```
lm(wage ~ experience + I(experience^2) + education + education*parttime,
    CPS1988)

##
## Call:
## lm(formula = wage ~ experience + I(experience^2) + education +
##      education * parttime, data = CPS1988)
##
## Coefficients:
##              (Intercept)              experience              I(experience^2)
##              -432.942                27.399                -0.399
##              education      parttimeyes  education:parttimeyes
```

```
##                58.378                167.071                -33.804
```

También es posible aplicar a los vectores alguna transformación directamente, como el logaritmo o raíz cuadrada (con `log()` y `sqrt()`). Hay que ser cuidados con los valores que toman las variables a las que se aplican estas funciones, recordando que la raíz cuadrada no está definida para números negativos y que  $\ln(0) \rightarrow -\infty$ :

```
sqrt(-1)
## [1] NaN
log(0)
## [1] -Inf
```

### 3.4 Extraer y calcular información de una regresión

Un objeto de clase `lm` —como `mincer`— es una lista que contiene varios atributos. Estos atributos son objetos calculados por `lm()` con información a la que podemos acceder. Para imprimir una lista completa de los atributos de un objeto usamos `attributes()`:

```
attributes(mincer)
## $names
## [1] "coefficients" "residuals" "effects" "rank"
## [5] "fitted.values" "assign" "qr" "df.residual"
## [9] "xlevels" "call" "terms" "model"
##
## $class
## [1] "lm"
```

Para usar algún atributo necesitamos la notación de signo `$`. Por ejemplo, podemos guardar el vector  $\hat{y}$  con

```
yhat <- mincer$fitted.values
```

Esto puede ser útil, por ejemplo, al implementar el método de variables instrumentales en dos etapas, donde tenemos que rescatar los valores estimados  $\hat{x}$  en la primera etapa para luego incluirlos en la ecuación original. Escribiendo `?lm` puedes obtener más información sobre todos estos atributos.

Vimos que `summary()` calcula e imprime otra información útil del resultado de una regresión. Análogo a `lm()`, `summary()` también tiene atributos a los que podemos acceder:

```
mincer.summ <- summary(mincer)
attributes(mincer.summ)
## $names
## [1] "call" "terms" "residuals" "coefficients"
## [5] "aliased" "sigma" "df" "r.squared"
## [9] "adj.r.squared" "fstatistic" "cov.unscaled"
##
```

```
## $class
## [1] "summary.lm"
```

Usando estos atributos podemos calcular fácilmente algunos resultados útiles. Por ejemplo, podemos obtener la matriz de varianzas-covarianzas del modelo:

```
(mincer.summ$sigma)^2 * mincer.summ$cov.unscaled
##              (Intercept) education experience
## (Intercept)      175.4      -11.09      -1.345
## education        -11.1       0.78       0.050
## experience        -1.3       0.05       0.038
```

Finalmente, además de los cálculos que podemos efectuar directamente con estos atributos, existen muchas funciones que toman un objeto de clase `lm` y calculan directamente algún resultado. Por ejemplo, podríamos haber calculado la matriz de varianzas-covarianzas directamente con `vcov()`:

```
vcov(mincer)
##              (Intercept) education experience
## (Intercept)      175.4      -11.09      -1.345
## education        -11.1       0.78       0.050
## experience        -1.3       0.05       0.038
```

Otras funciones útiles para usar con este tipo de objetos son:

```
# Calcular suma de los cuadrados de los residuos (SSR):
deviance(mincer)
# Extraer log-likelihood (LL):
logLik(mincer) # no tiene mucho sentido aquí porque no estimamos con MV
# Calcular el criterio de información de Akaike (AIC):
AIC(mincer)
```

Notar que `AIC()` define el criterio como

$$AIC = -2 \ln \hat{L}(p) + K \cdot 2, \quad (8)$$

donde  $K$  es el número de parámetros estimados y  $\hat{L}(p)$  es la función de verosimilitud. En muchos casos se prefiere que el criterio sea  $AIC/N$ , es decir, usar el Criterio de Información Bayesiano (BIC). Esto es,

$$BIC = -2 \ln \hat{L}(p) + K \cdot \ln N, \quad (9)$$

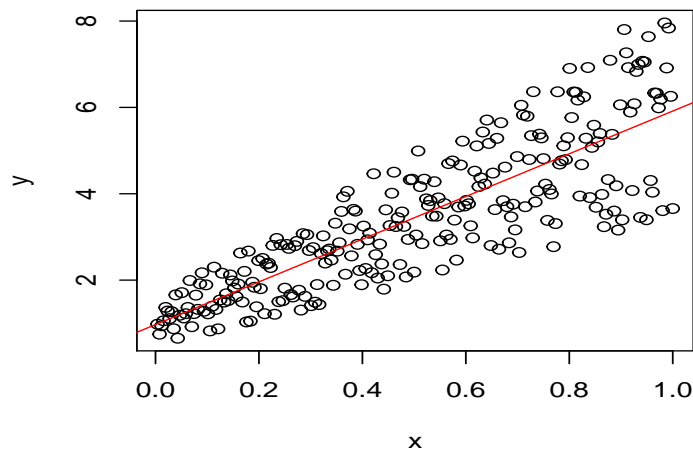
donde  $N$  es el número de observaciones. Para calcular el BIC también usamos `AIC()`, pero especificando un parámetro de penalización distinto:

```
AIC(mincer, k = log(NROW(CPS1988))) # fijar k=2 es igual que el AIC clásico
## [1] 418915
```

### 3.5 Heterocedasticidad

La heterocedasticidad es un problema que es más difícil de pronunciar que de entender: ocurre cuando la dispersión de la variable dependiente es cambiante para distintos valores de la variable independiente. Podemos generar fácilmente un conjunto de datos para visualizar el problema:

```
set.seed(314)
n <- 256                                # Número de observaciones
x <- (1:n)/n                             # Valores de `x`
e <- rnorm(n, sd=1)                      # Valores aleatorios de una normal
i <- order(runif(n, max=dnorm(e)))        # Ubicamos los más grandes al final
y <- 1 + 5 * x + e[rev(i)]               # Generamos `y` con el error `e`.
modelo <- lm(y ~ x)                      # Guardar modelo lineal
plot(x, y)
abline(coef(modelo), col = "Red")
```



En presencia de heterocedasticidad una estimación por MCO seguirá entregando coeficientes consistentes e insesgados. Sin embargo, no podremos estimar correctamente la matriz de varianzas-covarianzas, lo que producirá que los errores estándar de los coeficientes estén sesgados. Esto conduce a errores en tests de inferencia, como (por ejemplo) al determinar si un coeficiente es significativo.

#### 3.5.1 Test de Breusch-Pagan

El test de Breusch-Pagan es uno de los más clásicos para testear homocedasticidad. El estadístico se distribuye  $n\chi^2$  con  $k$  grados de libertad. La hipótesis nula corresponde a un modelo homocedástico, de forma que si el estadístico tiene un  $p$ -value menor a un límite razonable (ej.  $p < 0.05$ ) entonces rechazamos la nula y decimos que el modelo es heterocedástico.

Podemos usar `car::ncvTest()` para realizar el test de Breusch-Pagan original:

```
library(car)
ncvTest(modelo)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 68    Df = 1    p = 1.4e-16
```

Sabemos que `modelo` es heterocedástico (así lo construimos), y esto se confirma con un valor de  $p$  muy cercano a 0: tiene 15 ceros a la derecha del separador de decimales!.

También es posible calcular la versión “estudentizada” del test, propuesta por Koenker (1981). Esta versión más robusta que la original<sup>3</sup>, y puede ser calculada con `lmtest::bptest()`:

```
library(lmtest)
bptest(modelo)

##
## studentized Breusch-Pagan test
##
## data:  modelo
## BP = 70, df = 1, p-value <2e-16
```

### 3.5.2 Matriz de covarianza robusta

Como mencioné, en presencia de heterocedasticidad los coeficientes estimados por MCO se mantienen insesgados, pero su varianza estimada será incorrecta. Para calcular una matriz de covarianzas robusta a heterocedasticidad usamos la función `cars::hccm()` o `sandwich::vcovHC()`.

Por ejemplo, continuando con nuestro modelo guardado en `mincer`, podemos verificar que presenta heterocedasticidad y luego calcular la matriz de covarianzas corregida:

```
# Testear existencia de heterocedasticidad:
ncvTest(mincer)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 3135    Df = 1    p = 0

# Calcular matriz de covarianzas robusta:
hccm(mincer)

##              (Intercept) education experience
## (Intercept)      210.4    -14.479    -1.602
## education        -14.5     1.075     0.068
## experience        -1.6     0.068     0.050
```

`hccm()` aplica la llamada “corrección de White” para el cálculo de la matriz. Es interesante verificar

---

<sup>3</sup>Más detalles de las diferencias en <https://stats.stackexchange.com/a/193070/91358> y en Koenker (1981).

empíricamente una consecuencia importante a la hora de usar esta corrección: los errores robustos serán insesgados (que es lo que queremos), pero el costo es pérdida de eficiencia, es decir, errores más grandes. Compara el resultado anterior con la matriz que calculamos al usar `vcov(mincer)`.

También podemos usar `vcovHC()` para calcular la matriz. Esta función toma como método “base” la corrección de White, pero tiene disponibles otros métodos que son variaciones esa base. El método por defecto es HC3, que es equivalente a la opción `robust` del comando `regress` de Stata.<sup>4</sup>

Si no nos interesa la matriz de covarianzas en sí misma y sólo queremos obtener nuestros resultados con errores estándar robustos, podemos usar `summary()` con la opción `robust=TRUE`. Esta opción internamente usa `vcovHC()` con HC3 (ver nota al pie):

```
summary(mincer, robust = TRUE)
```

### 3.6 Tests de hipótesis lineal

El paquete `car` incluye la función `linearHypothesis()` para realizar tests de hipótesis lineal. Puede realizarse un test F o test de Wald, usando la matriz de covarianza regular o ajustada, dependiendo de nuestras especificaciones. Para realizar un test tenemos que construir una matriz de hipótesis y un vector de resultados. Por ejemplo, supongamos que tenemos un modelo lineal con 5 parámetros (incluyendo el intercepto):

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$$

y queremos testear

$$H_0 : \beta_0 = 0 \wedge \beta_2 + \beta_3 = 1.$$

En este caso la matriz de hipótesis y el vector de resultado serían

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \beta = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

En R implementamos esto de la siguiente forma:

```
modelo <- lm(y ~ x1 + x2 + x3 + x4)
mh <- rbind(c(1,0,0,0,0), c(0,0,1,1,0))
vr <- c(0,1)
linearHypothesis(modelo, mh, vr)
```

Si `modelo` es un objeto `lm` (como en este caso), se realizará un test  $F$  por defecto, mientras que si el objeto es `glm` entonces se realizará un test de Wald  $\chi^2$ . El tipo de test puede cambiarse por medio de la opción `type`. Asegúrate de revisar el documento de ayuda escribiendo `?linearHypothesis` para ver más detalles.

<sup>4</sup>Ver <https://stats.stackexchange.com/q/117052/91358> para una explicación breve sobre replicar el método de Stata. Long y Ervin (2000) conducen varias simulaciones para estudiar los distintos estimadores HC implementados en `vcovHC()`.

Podemos realizar un test de hipótesis usando una matriz de covarianzas robusta. La manera más simple de aplicar la corrección de White es con el argumento `white.adjust`:

```
linearHypothesis(modelo, mh, vr, white.adjust = TRUE)
```

En términos más generales, podemos pasar a `linearHypothesis()` la matriz de covarianzas que queramos (como las que calculamos en 3.5.2) por medio del argumento `vcov`. Por ejemplo, podríamos pasar una matriz de covarianzas con la corrección de Newey-West al test de hipótesis lineal escribiendo

```
linearHypothesis(modelo, mh, vr, vcov = NeweyWest(modelo))
```

Entonces `white.adjust = TRUE` es equivalente a escribir `vcov = hccm(modelo)`.

### 3.7 Mínimos cuadrados ponderados

Un modelo de mínimos cuadrados ponderados es una respuesta directa al problema de heterocedasticidad. En el modelo lineal simple asumimos que la variación del error es constante para todos los valores de las variables dependientes. Si este supuesto no se cumple, entonces hay puntos que en realidad entregan menos información sobre el proceso que estamos tratando de explicar, y parece razonable entonces penalizar estos puntos en la ponderación. Mínimos cuadrados ponderados es una implementación de esta idea, y permite maximizar la eficiencia de los parámetros estimados en presencia de heterocedasticidad.

Para realizar una estimación por mínimos cuadrados ponderados tenemos que pasar un vector con las ponderaciones al argumento `weights` de `lm()`:

```
modelo.mcp <- lm(smokes ~ 0 + male, data = smokerdata, weights = ponderaciones)
## Error in is.data.frame(data): object 'smokerdata' not found
```

### 3.8 Modelos con variables categóricas

Una variable categórica es cualquiera donde los valores no tienen valor numérico en sí mismo, si no que representan grupos o categorías. Por ejemplo, una variable que indica la región de Chile en la que se ubica una empresa es categórica. Cada región tiene un número asignado, pero dicho número solo codifica un significado; no el valor en sí no debe usarse como un número.

En R las variables categóricas se denominan factores. Puedes chequear que una variable sea factor con la función `is.factor()`:

```
is.factor(algo)
## Error in is.factor(algo): object 'algo' not found
```

Si una variable es un factor, entonces al incluirla en la regresión se producirá automáticamente un conjunto de dummies por cada nivel del factor.

En caso de que una variable no esté guardada como factor, puedes usar `factor()` para transformarla. Esto funciona dentro de la regresión también:

```
summary(lm(pc ~ emple + factor(region)))  
## Error in eval(predvars, data, env): object 'pc' not found
```

## 4 Datos de panel

Una base de datos de corte transversal (como las que hemos visto hasta ahora) posee una estructura interna unidimensional. Cada unidad (persona, empresa, familia) está indexada por  $i$ , y la base de datos es una colección de observaciones independientes. En un panel los datos poseen una estructura interna indexada por un arreglo de dos dimensiones: una dimensión transversal de unidades indexadas por  $i$ , y una dimensión temporal indexada por  $t$ . En términos intuitivos, un panel es equivalente a tener varias bases de corte transversal donde observamos a las mismas unidades en distintos períodos.

Cuadro 1: Ejemplo simple de la estructura de un panel

ID	Periodo	X
101	2015	6.5
102	2015	5.2
103	2015	5.9
101	2016	6.7
102	2016	5.5
103	2016	6.0

En estricto rigor la nueva dimensión del panel, indexada por  $t$ , no tiene que representar períodos. Por ejemplo, podríamos tener un panel donde observamos a los mismos estudiantes en distintas asignaturas, y de esa forma  $t$  sería un índice de asignatura. Sin embargo, en la gran mayoría de los casos se asume implícitamente que la nueva dimensión del panel representa períodos de algún tipo.

En el contexto de econometría, los modelos de efecto fijo y de efecto aleatorio son modelos de datos de panel que toman en cuenta la variación transversal de los datos. Sea  $i = 1, \dots, n$  el índice transversal y  $t = 1, \dots, T$  el índice temporal (o más en general, el índice que varía dentro de un grupo). Definimos un modelo de efecto fijo como

$$y_{it} = \alpha + u_i + \beta X_{it} + \epsilon_{it}. \quad (10)$$

Podemos entender este modelo, por ejemplo, como un panel de 30 estudiantes que son observados en tres períodos: 4<sup>to</sup> básico, 8<sup>vo</sup> básico y 4<sup>to</sup> medio. Cada estudiante está identificado por el subíndice  $i = 1, \dots, 30$  y cada período en el tiempo está identificado por el subíndice  $t = 1, 2, 3$ . Los valores que toman los indicadores no son relevantes, de forma que  $i$  podría haber sido una variable de ID de alumno y  $t$  podría haber sido una variable del grado del estudiante.



En este modelo cada unidad identificada por  $i$  tiene un intercepto que es invariante en el tiempo,  $\alpha + u_i$ . En términos del ejemplo, cada estudiante tiene un intercepto distinto. Es por esto que usualmente nos interesa solo incluir el efecto, pero los sigue interesando solamente  $\beta$ .

Un modelo de efecto aleatorio se define por la misma ecuación, pero se impone la restricción adicional de que el efecto específico a cada unidad no está correlacionado con las variables independientes  $X_{it}$ , es decir,  $E[u_i, X_{it}] = 0$ . En términos intuitivos esto corresponde a una versión más estricta del efecto fijo, ya que este último permite correlación entre dicho efecto y las variables independientes. Es lamentable que el nombre de “efecto fijo” y “efecto aleatorio” tengan poca relación con lo que realmente significa cada uno.

## 4.1 Efectos fijos

La manera más directa de definir un modelo de efecto fijo es incluyendo una dummy por cada unidad, es decir, haciendo que el indicador transversal  $i$  sea un factor.

```
lm(y ~ factor(index) + x)
## Error in unique.default(x, nmax = nmax): unique() applies only to vectors
```

Sin embargo esta solución puede resultar incómoda si el número de unidades es muy grande. Por ejemplo, si tenemos un panel de 3000 estudiantes entonces se calculará explícitamente dicha cantidad de coeficientes ( $u_i$ ), los cuales no nos interesan.

Otra manera común de estimar un modelo de efecto fijo es quitando el efecto fijo por la vía de estimar un modelo de diferencia de medias:

$$(y_{it} - \bar{y}_i) = \alpha + \beta(X_{it} - \bar{X}_i) + \zeta_{it}. \quad (11)$$

Al estimar (11) obtendremos el llamado *within estimator*, que indica la variación promedio a lo largo de  $t$  para cada grupo definido por  $i$

## Referencias

- Blackburn, McKinley y David Neumark (1992). “Unobserved Ability, Efficiency Wages, and Interindustry Wage Differentials”. En: *The Quarterly Journal of Economics* 107.4, págs. 1421-1436.
- Koenker, Roger (1981). “A note on studentizing a test for heteroscedasticity”. En: *Journal of Econometrics* 17.1, págs. 107-112.
- Long, J. Scott y Laurie H. Ervin (2000). “Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model”. En: *The American Statistician* 54.3, págs. 217-224.
- Wooldridge, Jeffrey M. (2013). *Introductory econometrics: a modern approach*. 5th ed. OCLC: ocn827938223. Mason, OH: South-Western Cengage Learning. 881 págs.