Ajuste a una linea recta

A. Zellner, J. Abellán

14 de febrero de 2018

Sigo el libro de Arnold Zellner, *An Introduction to Bayesian Inference in Econometrics*, epígrafe 3.1.2, página 60

El modelo: la linea recta

El modelo es bien sencillo y conocido

$$y = \beta_1 + \beta_2 * x$$

Datos

Utilizaremos como ejemplo datos generados al azar de acuerdo con el modelo

$$y_i = \beta_1 + \beta_2 * x_i + \epsilon_i \ , \ i = 1, 2, \cdots, N_d$$

donde $\epsilon \sim N(0, \sigma)$ sigue una normal de media cero y desviación σ

```
# los datos
Nd <- 10

#variable de control
x <- 1 : Nd

#simulación del ruido experimental
sigma <- .1

#ordenada en el origen verdadera
y0 <- .45

#pendiente verdadera
pendiente <- .13

# datos generados
y <- y0 + pendiente * x + rnorm( Nd, 0, sigma )

#Nd <- length( x )
```

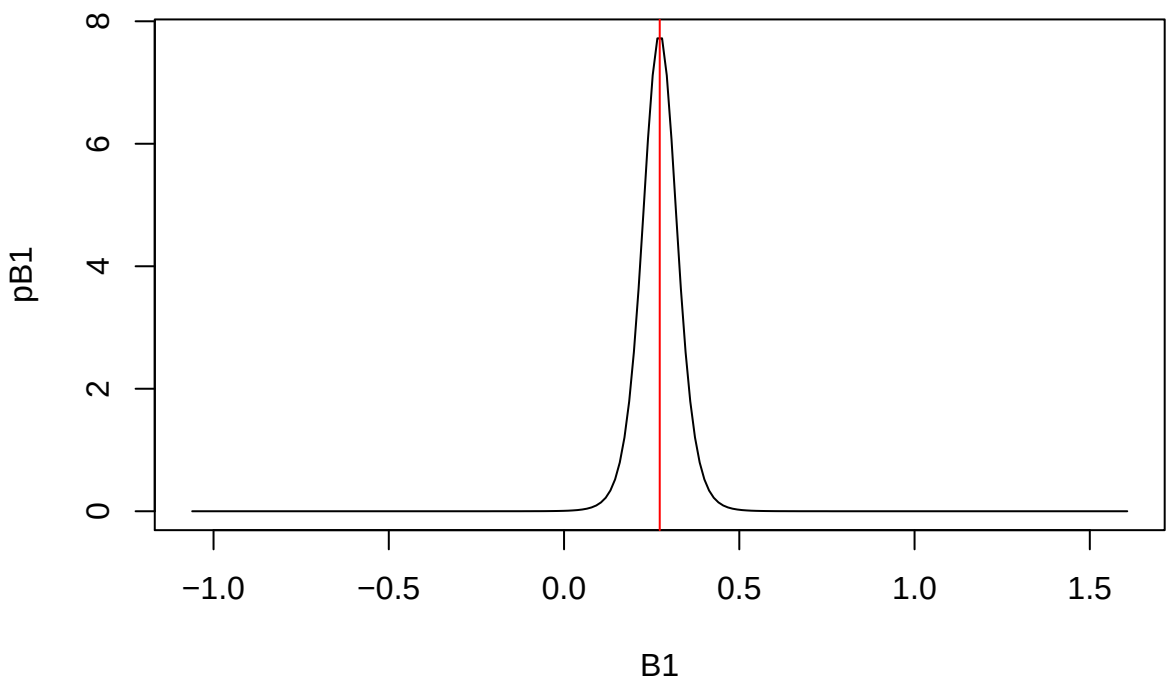
```
# los estadísticos
xm <- mean( x )

ym <- mean( y )

x2m <- mean(x^2)

sx2 <- var( x )

sxy <- cov( x, y )
```

```r
sy2 <- var( y )

sy <- sd( y )

# la pendiente óptima
b2op <- sxy / sx2

b2opr <- round( b2op, 3 )

#la ordenada en el origen óptima
b1op <- ym - b2op * xm

b1opr <- round( b1op, 3 )

# suma de los cuadrados de los residuos
Qmin <- sum( ( y - b1op - b2op * x )^2 )

# rango de los parámetros
rangoY <- abs( max( y ) - min( y ) )

rangoP <- max( y ) - min( x )

b1min <- b1op - rangoY ; b1max <- b1op + rangoY

b2min <- b2op - rangoP / 4 ; b2max <- b2op + rangoP / 4

smin <- sy / 100 ; smax <- sy * 1.2

# discretización del espacio de parámetros
n <- 200

B1 <- seq( b1min, b1max, len = n ) ; dB1 <- ( b1max - b1min ) / ( n - 1 )

B2 <- seq( b2min, b2max, len = n ) ; dB2 <- ( b2max - b2min ) / ( n - 1 )

SGM <- seq( smin , smax, len = n ) ; dSGM <- ( smax - smin ) / ( n - 1 )

# Solución teórica según Bayes:
# Véase epígrafe '4.2.1 Incertidumbre en el eje Y' del fichero
# ResumenProbabilidad2018.pdf
pB1 <- ( 1 + ( Nd / Qmin ) * ( sx2 / x2m ) * ( B1 - b1op )^2 )^(- ( Nd - 1 ) / 2 )

pB2 <- ( 1 + ( Nd / Qmin ) * sx2 * ( B2 - b2op )^2 )^(- ( Nd - 1 ) / 2 )

pSGM <- SGM^(-(Nd-1)) * exp( - Qmin / ( 2 * SGM^2 ) )

# sgm más probable
sgm0 <- sqrt( Qmin / ( Nd - 1 ) )

# Normalizamos las funciones de distribución
pB1 <- pB1 / sum( pB1 * dB1)

pB2 <- pB2 / sum( pB2 * dB2)
```

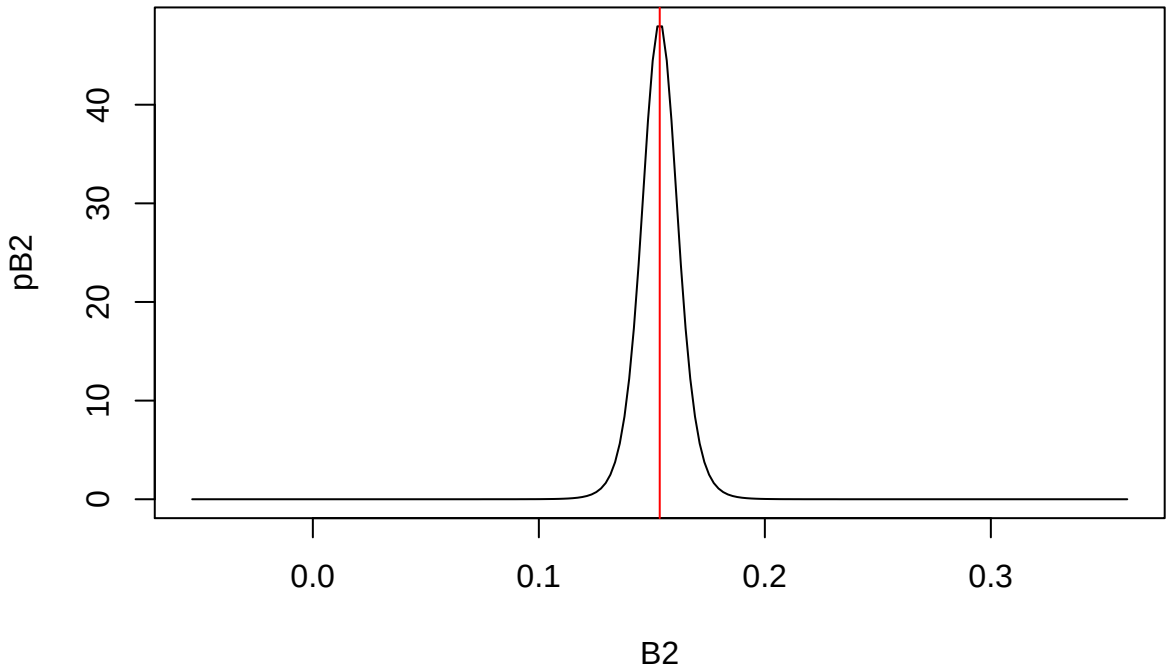
```r
pSGM <- pSGM / sum( pSGM * dSGM)

# función para calcular intervalos creíbles
IC <- function( B, pB, alfa ) {

    # índice que maximiza pB
    io <- which.max( pB )

    area <- i <- 0

    while ( area < alfa ) {

        i <- i + 1

        # ensanchamos el intervalo simétricamente
        imin <- io - i; imax <- io + i

        area <- sum( pB[ imin : imax ] * dB1 )

    }

    icB <- ( B[ imax ] - B[ imin ] ) / 2

    #icB <- round( icB, 3 )

}

# intervalos del 68% (aproximadamente)
deltaB1 <- IC( B1, pB1, 0.68 )

deltaB2 <- IC( B2, pB2, 0.68 )
```

Funciones de distribución

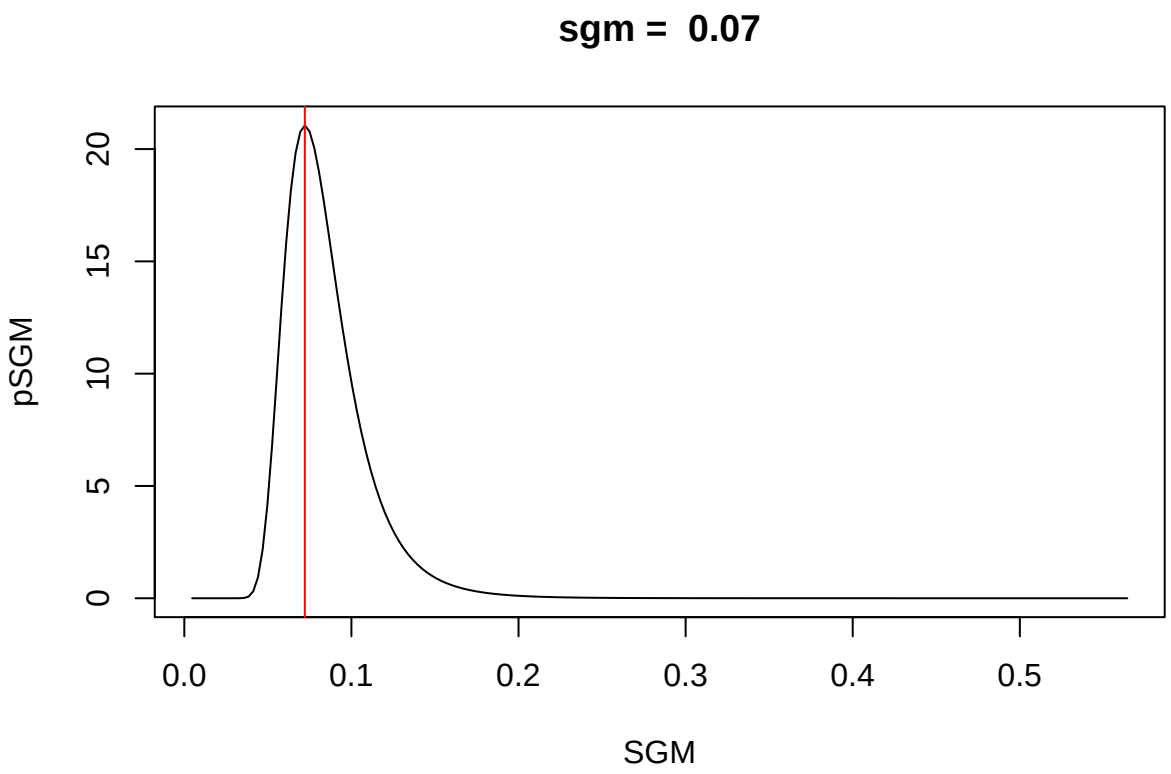
```r
plot( B1, pB1,

      type = "l",

      main = paste( "b1 = ", b1opr, "+-", round( deltaB1, 3 ) )

    )

abline( v = b1op, col = 2 )
```




```
plot( B2, pB2,
      type = "l",
      main = paste( "b2 = ", b2opr, "+-", round( deltaB2, 3 ) )
    )
abline( v = b2op, col = 2 )
```

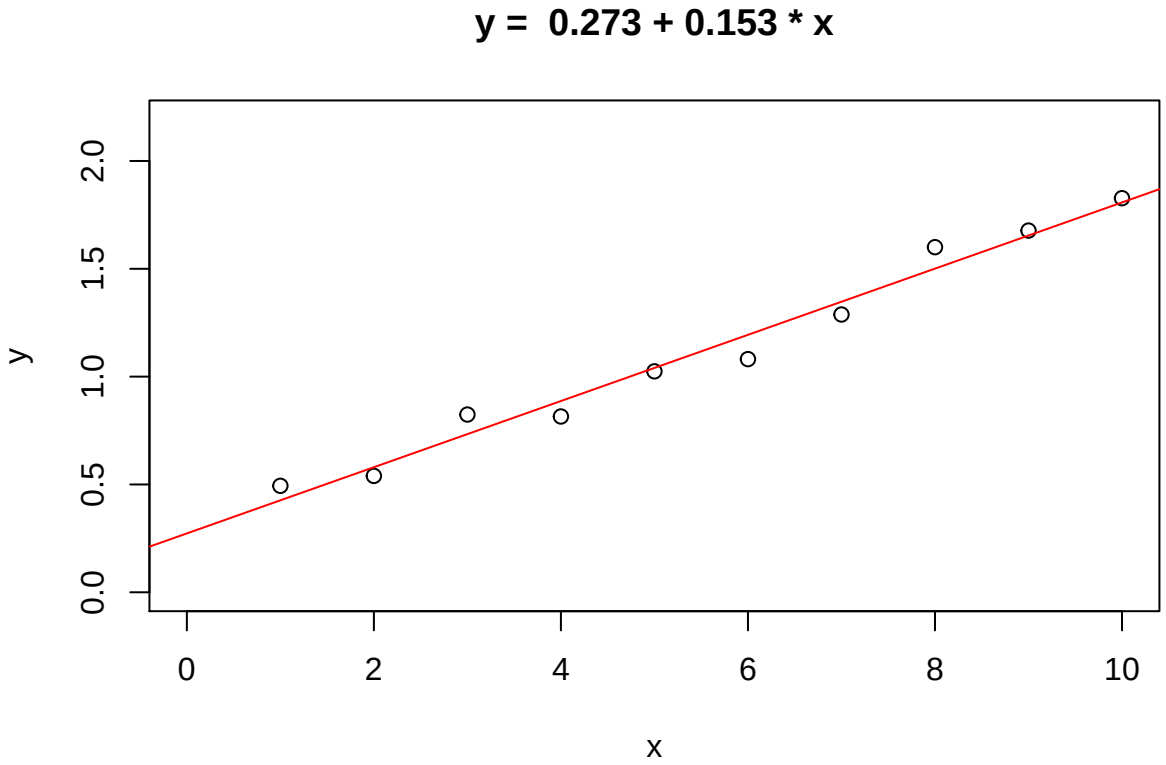



```
plot( SGM, pSGM,
      type = "l",
      main = paste( "sgm = ", round( sgm0, 2 ) )
      )
abline( v = sgm0, col = 2 )
```



El ajuste a los datos

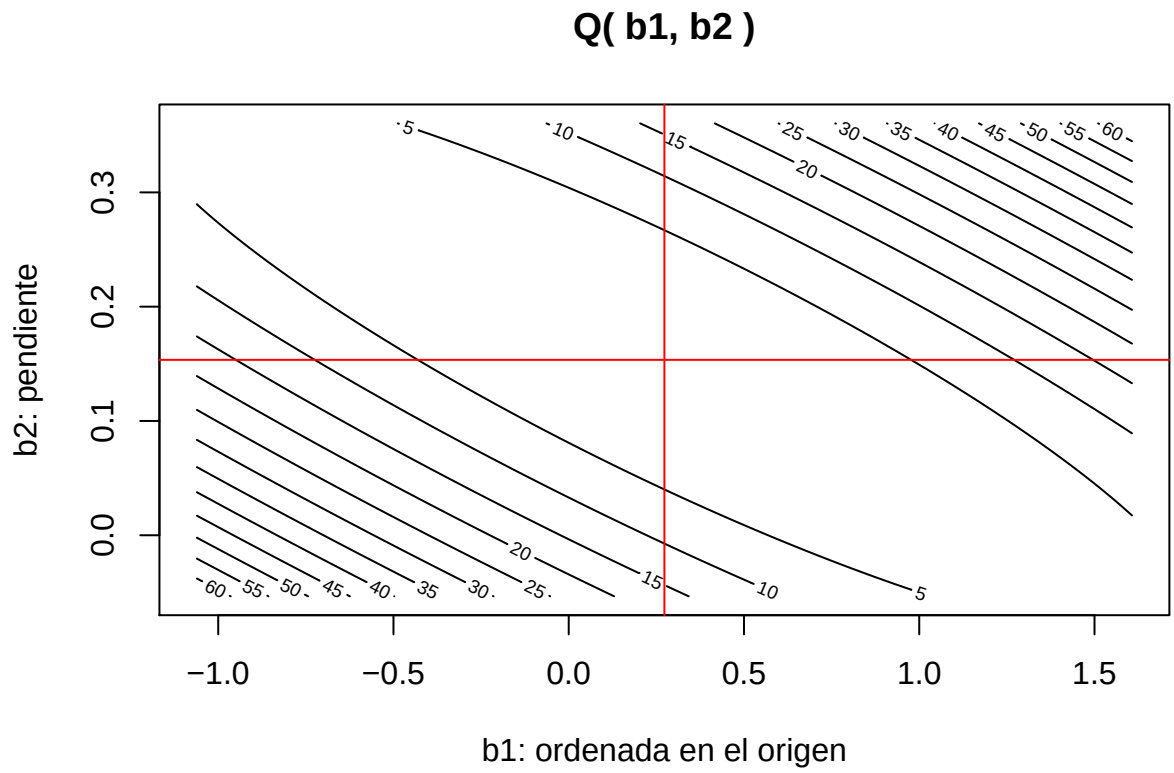
```
plot( x, y,

      xlim = c( 0, max( x ) ),

      ylim = c( min( 0, min( y ) ), 1.2 * max( y ) ),

      main = paste( "y = ", b1opr, "+", b2opr, "* x" )

    )

#abline( lm( y ~ x ), col = 3 )
abline( b1op, b2op, col = 2 )
```



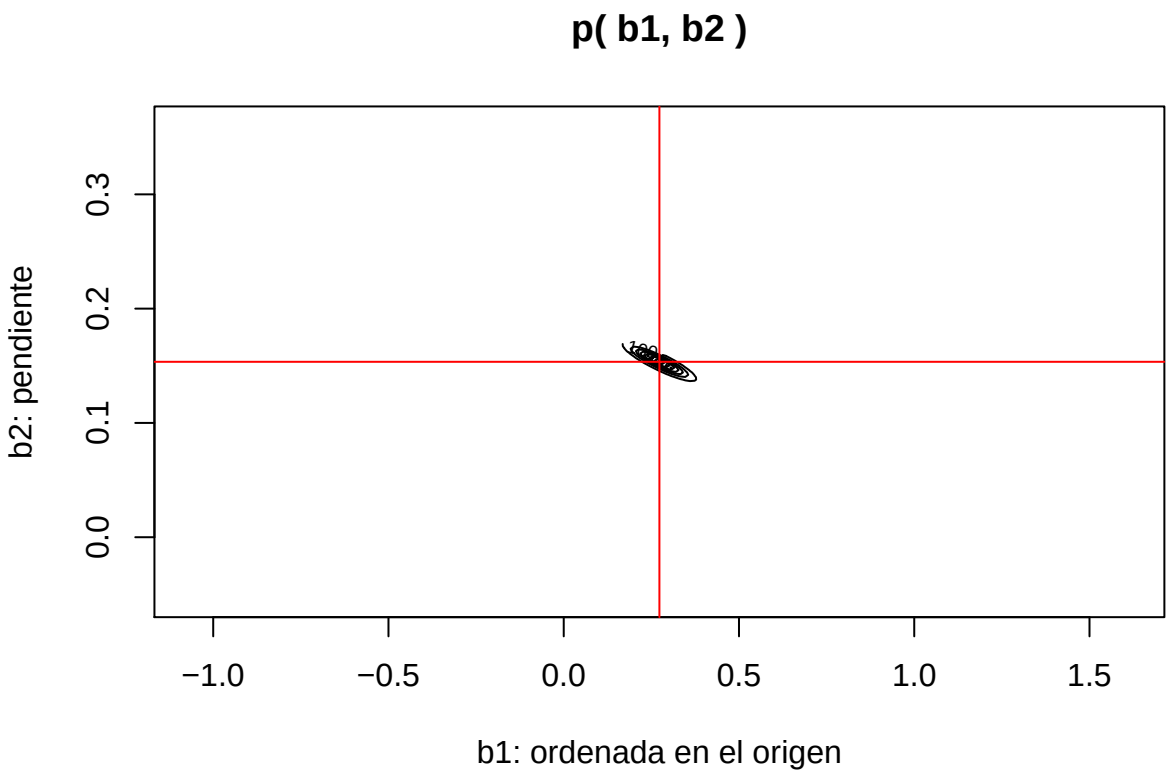
Si no somos capaces de obtener las soluciones analíticas de las diferentes funciones de distribución, siempre podemos recurrir al cálculo numérico y calcular la función $Q(\beta_1, \beta_2)$ y a partir de aquí $p(\beta_1, \beta_2)$

```
# funciones de distribución

Q <- matrix( 0, nrow = n, ncol = n )

# Cálculo de los elementos de Q
for (i in seq( along = B1 ) ) {

  for (j in seq( along = B2 ) ) {

    yteorica <- B1[ i ] +  B2[ j ] * x

    Q[ i, j ] <- sum( ( y - yteorica )^2 )

  }

}

# Cálculo del mínimo de Q
Qminn <- min( Q )

#curvas de nivel de la función Q
contour( B1, B2, Q,

         xlab = "b1: ordenada en el origen",

         ylab = "b2: pendiente",
```

```
         main = "Q( b1, b2 )"

        )

abline( v = b1op, h = b2op, col = 2 )
```




```
# p(b1, b2)
pB1B2 <- Q^( - Nd / 2 )

# normalizamos
pB1B2 <- pB1B2 / sum( pB1B2 * dB1 * dB2 )

# curvas de nivel de p(b1, b2)
contour( B1, B2, pB1B2,

         xlab = "b1: ordenada en el origen",

         ylab = "b2: pendiente",

         main = "p( b1, b2 )"

        )

abline( v = b1op, h = b2op, col = 2 )
```


p(b1, b2)
b2: pendiente
0.0
0.1
0.2
0.3
−1.0
−0.5
0.0
0.5
1.0
1.5
b1: ordenada en el origen