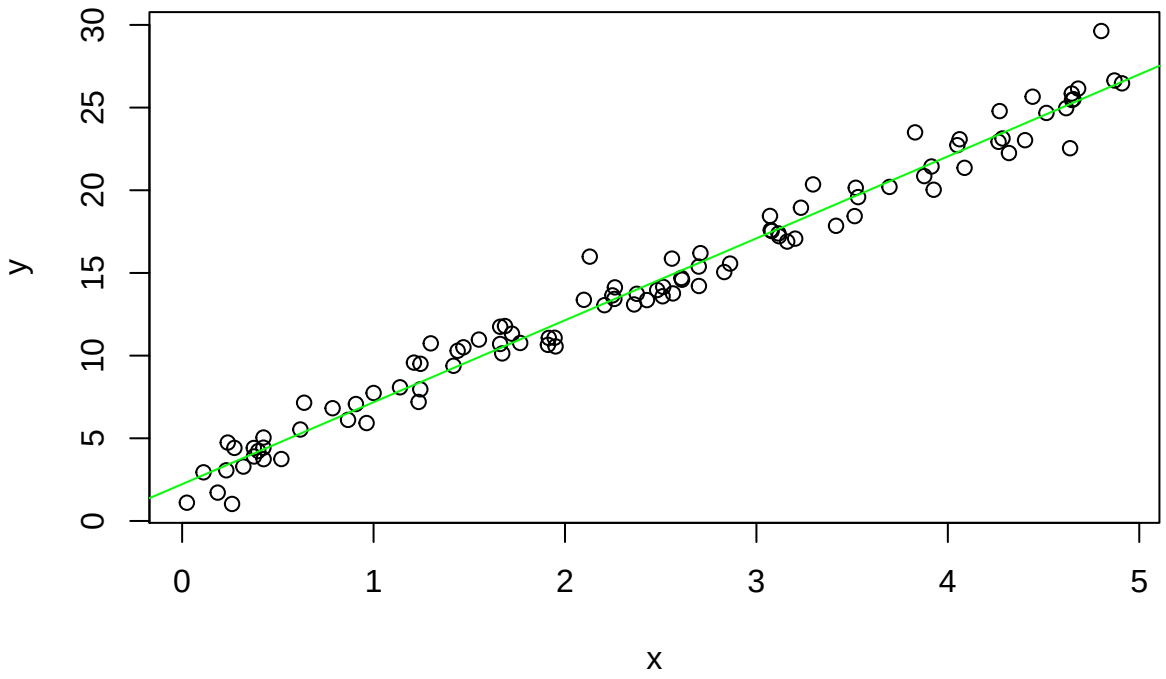
Practica3_Apartado2.R

moises

Fri Jun 16 07:05:07 2017

```
############################
##########Apartado2##########
############################

#Le vuelvo a enviar esta práctica para corregir el siguiente error que me indicó:

#En cuanto al segundo boletín, tu calificación es de 1.25 porque no están bien definidos
#los input de las funciones. Muchas de las funciones que has creado (como f,
#sumatorioBeta0,  sumatorioBeta1, descenso_gradiente,...) utilizan valores de x e y que,
#sin embargo, no son pasados como argumentos. El código se ejecuta sin errores porque has
#definido fuera x e y antes de compilar pero, en caso de cambiar de muestra, debería
#recompilar de nuevo todas las funciones para que se ejecutase sin problemas.

#He corregido dicho error para que tanto x como y sean pasados como argumentos en todas
#las funciones.

n <- 100
x <- runif(n, min = 0, max = 5) # x_i: n puntos aleatorios en el itervalo [min,max]
beta0 <- 2 # Parámetro beta0 del modelo
beta1 <- 5 # Parámetro beta1 del modelo
epsilon <- rnorm(n, sd = 1) # error (con desviación típica sd=1)
y <- beta0 + beta1 * x + epsilon # y_i

#Ploteamos
plot(x,y, main="Diagrama de Dispersión") #Diagrama de dispersión
abline(lm(y ~ x), col="green")
```



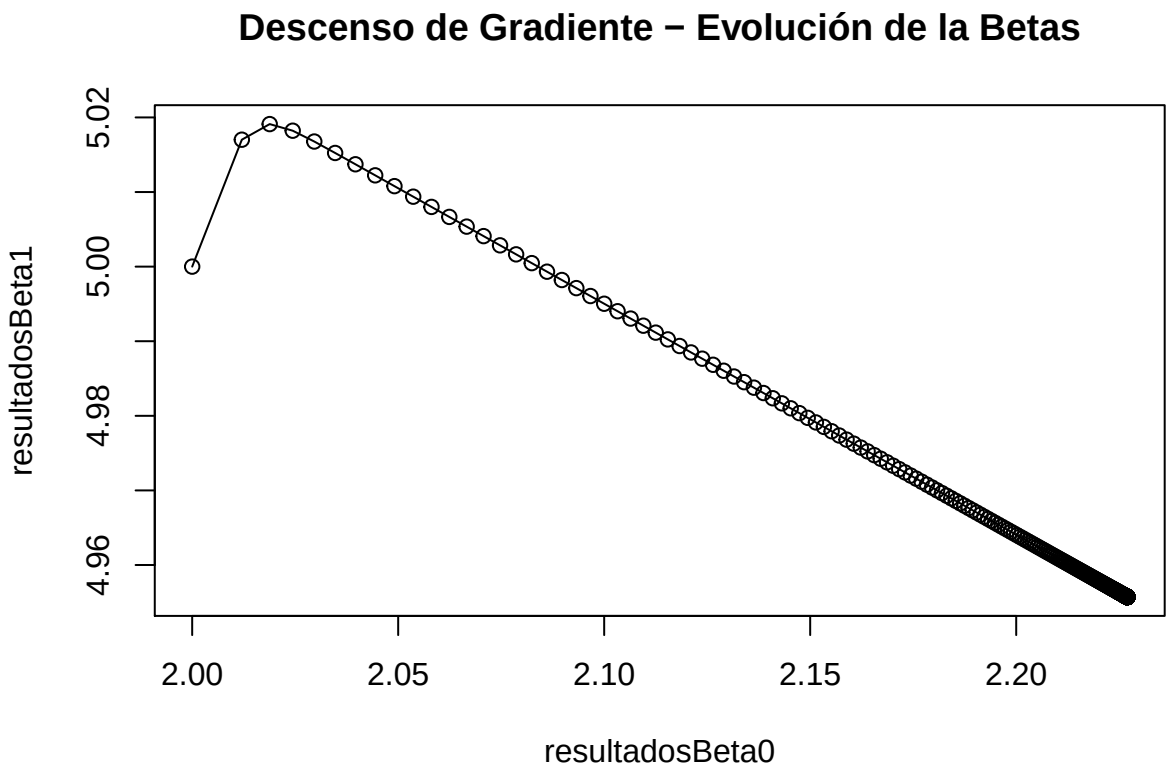

```
#Función a Minimizar
f <- function(x.parametro, y.parametro, beta0.parametro, beta1.parametro) {
  suma = 0
  for(i in 1:100){suma = suma +
      (y.parametro[i] - beta0.parametro - beta1.parametro * x.parametro[i])^2}
  0.5*suma
}

mod <- lm(y ~ x) #Regresión de la recta de entrenamiento
coef(mod) #Coeficientes
```

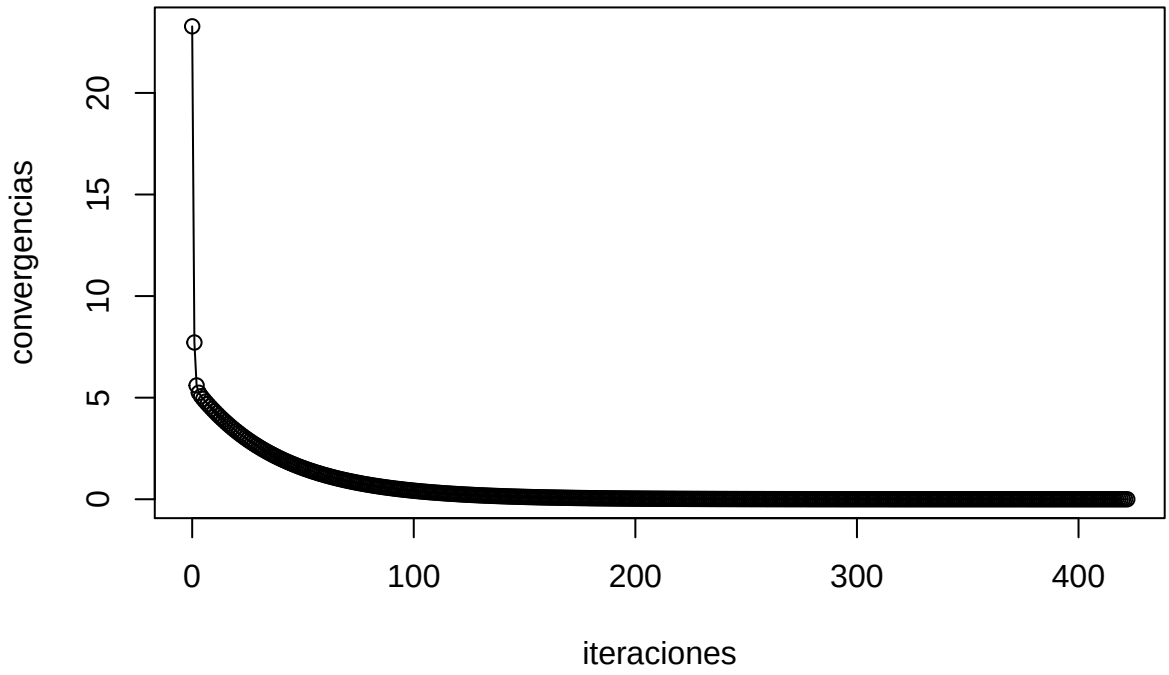
```
## (Intercept)           x
##    2.226966    4.955710
```

```
#Realizamos la derivada a la función a minimizar
#Y obtenemos (y - beta0 - beta1 * x) + x * (y - beta0 - beta1 * x)

#Por lo que hacemos la función del sumatorio para el primer parámetro beta0 sombrero
#con el primer término de la derivada
sumatorioBeta0 <- function(x.parametro, y.parametro, beta0.parametro, beta1.parametro) {
  suma <- 0
  for(i in 1:n) {suma <- suma + (y.parametro[i] - beta0.parametro - beta1.parametro *
                                                                    x.parametro[i])}
  return(suma)
}

#Y hacemos la función del sumatorio para el segundo parámetro beta1 sombrero
#con el segundo término de la derivada
sumatorioBeta1 <- function(x.parametro, y.parametro, beta0.parametro, beta1.parametro) {
  suma <- 0
```

```
  for(i in 1:n) {suma <- suma + x.parametro[i] * (y.parametro[i] - beta0.parametro -
                                                    beta1.parametro * x.parametro[i])}
  return(suma)
}

#Función que calcula la norma vectorial de un vector dado
norma_vectorial <- function(x.parametro) {
  suma <- 0
  for(i in 1:length(x.parametro)) {
    suma = suma + x.parametro[i]^2
  }
  return(sqrt(suma))
}

#Ahora ya podemos hacer el algoritmo del descenso del gradiente
descenso_gradiente <- function(x.parametro, y.parametro, b0.parametro, b1.parametro,
                               t.parametro,tol.parametro,lim.parametro) {
  b0.algoritmo = b0.parametro
  b1.algoritmo = b1.parametro
  i <- 0
  iteraciones <- c(i)
  resultadosBeta0 <- c(b0.algoritmo)
  resultadosBeta1 <- c(b1.algoritmo)
  convergencias <- c(norma_vectorial(c(sumatorioBeta0(x.parametro, y.parametro,
                                                      b0.algoritmo,b1.algoritmo),
                                       sumatorioBeta1(x.parametro, y.parametro,
                                                      b0.algoritmo,b1.algoritmo))))
  while (norma_vectorial(c(sumatorioBeta0(x.parametro, y.parametro, b0.algoritmo,
                                          b1.algoritmo),
                           sumatorioBeta1(x.parametro, y.parametro, b0.algoritmo,
                                          b1.algoritmo)))
                           > tol.parametro & i < lim.parametro) {
    b0.algoritmo = b0.algoritmo + t.parametro*sumatorioBeta0(x.parametro, y.parametro,
                                                             b0.algoritmo,b1.algoritmo)
    b1.algoritmo = b1.algoritmo + t.parametro*sumatorioBeta1(x.parametro, y.parametro,
                                                             b0.algoritmo,b1.algoritmo)
    i <- i + 1
    iteraciones = append(iteraciones,i)
    resultadosBeta0 = append(resultadosBeta0,b0.algoritmo)
    resultadosBeta1 = append(resultadosBeta1,b1.algoritmo)
    convergencias = append(convergencias,
                           norma_vectorial(c(
                                 sumatorioBeta0(x.parametro, y.parametro,
                                                b0.algoritmo, b1.algoritmo),
                                 sumatorioBeta1(x.parametro, y.parametro,
                                                b0.algoritmo,b1.algoritmo))))
    #print(norma_vectorial(c(sumatorioBeta0(x.parametro, y.parametro,
    #                        b0.algoritmo,b0.algoritmo),
    #                      sumatorioBeta1(x.parametro, y.parametro,
    #                        b0.algoritmo,b0.algoritmo))))
  }
  plot(resultadosBeta0,resultadosBeta1,
       main = "Descenso de Gradiente - Evolución de la Betas")
```

```
  lines(resultadosBeta0,resultadosBeta1)
  plot(iteraciones,convergencias,
       main = "Descenso de Gradiente - Convergencia")
  lines(iteraciones,convergencias)
  cat("Número de Iteraciones =", as.character(i),"\n")
  c(b0.algoritmo,b1.algoritmo)
}

set.seed(1234) #Establecemos esta semilla para que aparezca siempre el mismo resultado
t <- 0.001
tolerancia <- 0.0001
limite <- 1000
beta.estimada <- descenso_gradiente(x, y, beta0, beta1, t, tolerancia, limite)
```






```
## Número de Iteraciones = 422
```

```
beta.estimada[1] #Resultado para Beta0 Sombrero
```

```
## [1] 2.226962
```

```
beta.estimada[2] #Resultado para Beta1 Sombrero
```

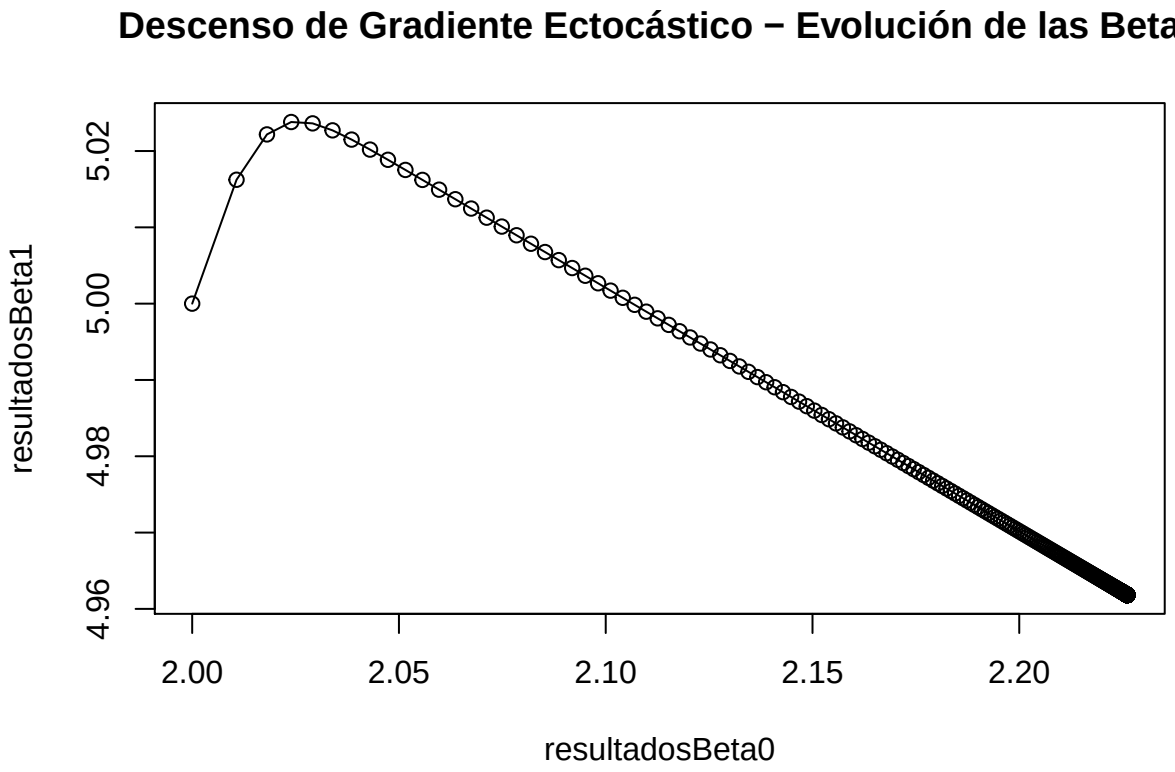
```
## [1] 4.955712
```

```
coef(mod) #Comparamos el resultado con los coeficientes obtenidos con lm
```

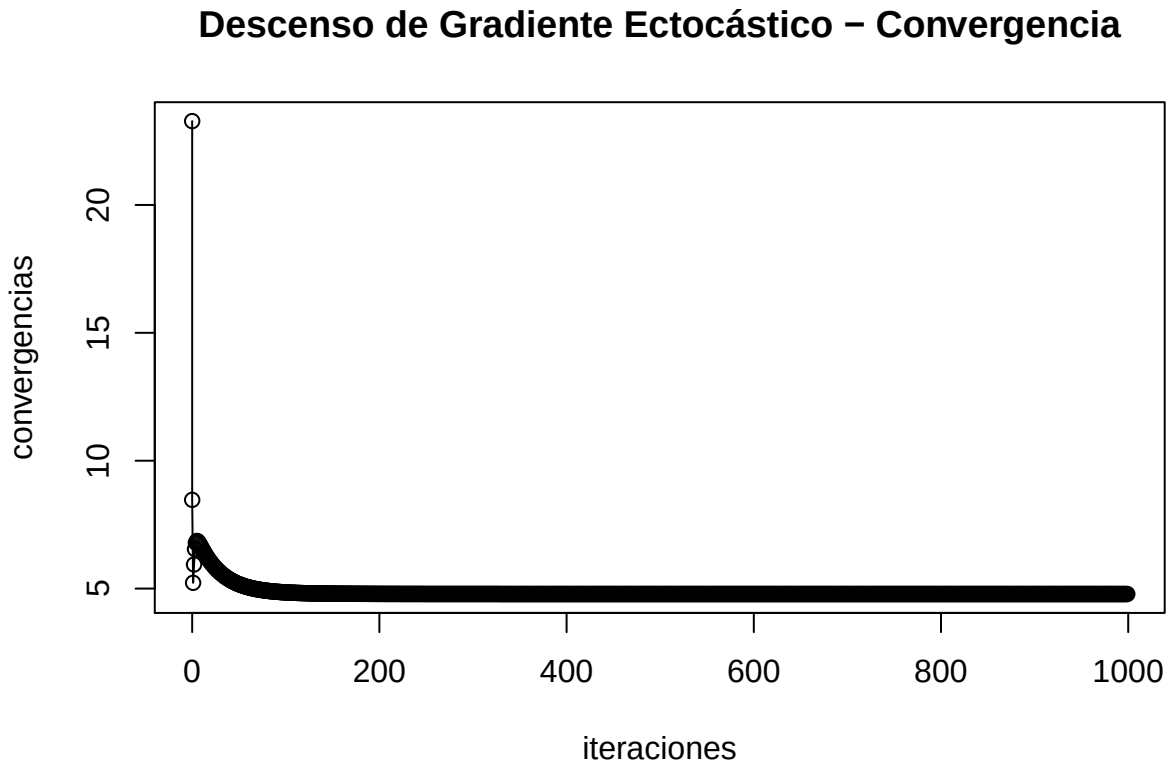
```
## (Intercept)           x
##    2.226966    4.955710
```

```
#############################################################################################

#Método Ectocástico
metodo_ectocastico <- function(x.parametro, y.parametro, b0.parametro,
                               b1.parametro,t.parametro,tol.parametro,lim.parametro) {
  b0.algoritmo = b0.parametro
  b1.algoritmo = b1.parametro
  j <- 0
  iteraciones <- c(j)
  resultadosBeta0 <- c(b0.algoritmo)
  resultadosBeta1 <- c(b1.algoritmo)
  convergencias <- c(norma_vectorial(c(
    sumatorioBeta0(x.parametro, y.parametro, b0.algoritmo,b1.algoritmo),
    sumatorioBeta1(x.parametro, y.parametro, b0.algoritmo,b1.algoritmo))))
  while (norma_vectorial(c(x.parametro, y.parametro,
         sumatorioBeta0(x.parametro, y.parametro, b0.algoritmo,b1.algoritmo),
        sumatorioBeta1(x.parametro, y.parametro, b0.algoritmo,b1.algoritmo))) >
```

```
        tolerancia & j < limite) {
    for(i in 1:n) {
      b0.algoritmo = b0.algoritmo + t*(y.parametro[i] - b0.algoritmo - b1.algoritmo *
                                       x.parametro[i])
      b1.algoritmo = b1.algoritmo + t*x.parametro[i] * (y.parametro[i] - b0.algoritmo -
                                                        b1.algoritmo * x.parametro[i])
    }
    iteraciones = append(iteraciones,j)
    resultadosBeta0 = append(resultadosBeta0,b0.algoritmo)
    resultadosBeta1 = append(resultadosBeta1,b1.algoritmo)
    convergencias = append(convergencias,
                           norma_vectorial(c(
                              sumatorioBeta0(x.parametro, y.parametro,
                                             b0.algoritmo,b1.algoritmo),
                              sumatorioBeta1(x.parametro, y.parametro,
                                             b0.algoritmo,b1.algoritmo))))
    j <- j + 1
    #print(norma_vectorial(c(sumatorioBeta0(x.parametro, y.parametro,
    #                          b0.algoritmo,b1.algoritmo),
    #                        sumatorioBeta1(x.parametro, y.parametro,
    #                          b0.algoritmo,b1.algoritmo))))
  }
  plot(resultadosBeta0,resultadosBeta1,
       main = "Descenso de Gradiente Ectocástico - Evolución de las Betas")
  lines(resultadosBeta0,resultadosBeta1)
  cat("Número de Iteraciones =", as.character(j),"\n")
  plot(iteraciones,convergencias,
       main = "Descenso de Gradiente Ectocástico - Convergencia")
  lines(iteraciones,convergencias)
  c(b0.algoritmo,b1.algoritmo)
}

set.seed(1234) #Establecemos esta semilla para que aparezca siempre el mismo resultado
t <- 0.001
tolerancia <- 0.0001
limite <- 1000
beta <- metodo_ectocastico(x, y, beta0, beta1, t, tolerancia, limite)
```




```
## Número de Iteraciones = 1000
```




```
beta[1] #Resultado para Beta0 Sombrero
```

```
## [1] 2.22613
```

```r
beta[2] #Resultado para Beta1 Sombrero
```

```
## [1] 4.96183
```

```r
coef(mod) #Comparamos el resultado con los coeficientes obtenidos con lm
```

```
## (Intercept)           x
##    2.226966    4.955710
```