

moodle_3_04-06-13-18

Alan Coila Bustinza

2022-06-04

```
library(knitr)      # For knitting document and include_graphics function
library(ggplot2)    # For plotting
library('png')
```

pregunta 1

```
img1_path <- "p1_2022-06-04_132015.png"
include_graphics(img1_path)
```

Dado el siguiente conjunto de puntos:

$$\begin{bmatrix} X & 0 & 3 & 6 & 9 & 10 \\ Y & -1 & -2 & 4 & -1 & 2 \end{bmatrix}$$

¿Cuáles son los valores de c_0 , c_1 y c_2 del polinomio de regresión $y = c_2 \cdot x^2 + c_1 \cdot x + c_0$?

NOTACIÓN: Escribe la respuesta con el formato $\{c_0 = A, c_1 = B, c_2 = C\}$

Respuesta:

$\{c_0 = -1.67412076, c_1 = 0.79782476, c_2 = -0.05295792\}$

La respuesta correcta es: $\{c_0 = -\frac{8235}{4919}, c_1 = \frac{7849}{9838}, c_2 = -\frac{521}{9838}\}$

```
x1 <- c(0,3,6,9,10)
y1 <- c(-1,-2,4,-1,2)
```

```

grado=2 # usualment las comparacions son lineales

myPhi <- function(x, n) {
  Phi <- matrix(1, length(x), n + 1)
  for (i in 1:n) {
    Phi[, i + 1] <- x^i # funciones base para el ajuste polinómico, segun el grado
  }
  return(Phi)
}

mylssolve <- function(A, y) {
  AT <- t(A)
  return((solve(AT %*% A)) %*% AT %*% y) # la función solve nos devuelve la inversa de la matriz
}

mypolyfit <- function(x, y, n) {
  Phi <- myPhi(x, n) # construimos la matriz con myPhi
  c <- mylssolve(Phi, y) # resolvemos el sistema de ecuaciones normales con la función mylssolve
  return(c)
}

myeval <- function(x, c) {
  f <- 0
  for (i in 1:length(c)) {
    f <- f + c[i] * x^(i - 1)
  }
  return(f)
}

mypolyfit(x1,y1,grado)

```

```

##           [,1]
## [1,] -1.67412076
## [2,]  0.79782476
## [3,] -0.05295792

```

pregunta 2

```

img1_path <- "p2_2022-06-04_132116.png"
include_graphics(img1_path)

```

Dado el siguiente conjunto de puntos:

$$\begin{bmatrix} X & 0 & 3 & 4 & 7 & 8 \\ Y & -1 & -5 & -1 & -2 & 0 \end{bmatrix}$$

¿Cuáles son las ecuaciones normales que determinan la recta de regresión $y = c_1 \cdot x + c_0$?

Seleccione una:

- ☐ a. $\{8 \cdot c_0 + 25 \cdot c_1 = 22, 25 \cdot c_0 + 139 \cdot c_1 = -33\}$
- ☒ b. $\{5 \cdot c_0 + 22 \cdot c_1 = -9, 22 \cdot c_0 + 138 \cdot c_1 = -33\}$ ✓
- Correcto!
- ☐ c. $\{5 \cdot c_0 + 19 \cdot c_1 = -33, 25 \cdot c_0 + 139 \cdot c_1 = -159\}$

Respuesta correcta

La respuesta correcta es: $\{5 \cdot c_0 + 22 \cdot c_1 = -9, 22 \cdot c_0 + 138 \cdot c_1 = -33\}$

```
x2 <- c(0,3,4,7,8)
y2 <- c(-1,-5,-1,-2,0)
grado <- 1

myPhi <- function(x, n) {
  Phi <- matrix(1, length(x), n + 1)
  for (i in 1:n) {
    Phi[, i + 1] <- x^i # funciones base para el ajuste polinómico, segun el grado
  }
  return(Phi)
}

ec_normal <- function(x,y,n){
  A <- myPhi(x,n)
  B <- t(A)%*%A
  C <- t(A)%*%y
  return (cbind(B,C))
}

ec_normal(x2,y2,grado)
```

```
##      [,1] [,2] [,3]
## [1,]    5   22   -9
## [2,]   22  138  -33
```

pregunta 3

```
img1_path <- "p3_2022-06-04_132205.png"
include_graphics(img1_path)
```

Dado este conjunto de puntos:

$$\begin{bmatrix} X & 1 & 2 & 5 & 7 & 10 \\ Y & 9 & 2 & 11 & 8 & 12 \end{bmatrix}$$

Calcula la covarianza entre los valores de X y los de Y.

Respuesta:

8.5

La respuesta correcta es: $\frac{17}{2}$

```
x3 <- c(1,2,5,7,10)
y3 <- c(9,2,11,8,12)
cov(x3,y3)
```

```
## [1] 8.5
```

pregunta 4

```
img1_path <- "p4_2022-06-04_132250.png"
include_graphics(img1_path)
```

Dado este conjunto de puntos de una cierta función:


$$\begin{bmatrix} X & 1 & 3 & 6 & 8 & 9 \\ Y & 17 & 12 & 6 & 12 & 11 \end{bmatrix}$$

Usa la recta de regresión para dar una aproximación del valor de esta función cuando x valga 20

Respuesta:

$\frac{\square}{\square}$ \square^\square $\sqrt{\square}$ $\sqrt[\square]{\square}$ (\square) $(\begin{smallmatrix} \square & \square \\ \square & \square \end{smallmatrix})$ \div π α \leftarrow \rightarrow $?$

2.168142



La recta de regresión es $-\frac{73}{113} \cdot x + \frac{1705}{113}$. Substituyendo x por 20 podemos encontrar una aproximación de la función en este punto.

La respuesta correcta es: $\frac{245}{113}$

```
x4 <- c(1,3,6,8,9)
y4 <- c(17,12,6,12,11)
grado=1 # recta de regresion

myPhi <- function(x, n) {
  Phi <- matrix(1, length(x), n + 1)
  for (i in 1:n) {
    Phi[, i + 1] <- x^i # funciones base para el ajuste polinómico, segun el grado
  }
  return(Phi)
}

mylssolve <- function(A, y) {
  AT <- t(A)
  return((solve(AT %*% A)) %*% AT %*% y) # la función solve nos devuelve la inversa de la matriz
}

mypolyfit <- function(x, y, n) {
  Phi <- myPhi(x, n) # construimos la matriz con myPhi
  c <- mylssolve(Phi, y) # resolvemos el sistema de ecuaciones normales con la función mylssolve
  return(c)
}

myeval <- function(x, c) {
  f <- 0
  for (i in 1:length(c)) {
```

```
    f <- f + c[i] * x^(i - 1)
  }
  return(f)
}
```

```
cof <- mypolyfit(x4,y4,grado)
cof
```

```
##           [,1]
## [1,] 15.0884956
## [2,] -0.6460177
```

```
myeval(20,cof)
```

```
## [1] 2.168142
```