Ejercicio

Uriel Paluch

27/11/2021

```
DerivadaPorDefinicion <- function(x, fx){
  fprima <- rep(NA, times = length(x))

  #Se asume que todos los valores estan separados por un h constante
  h <- x[2] - x[1]

  #Diferencia progresiva
  for (i in 1:(length(x)-1)){
    fprima[i] <- (fx[i+1] - fx[i]) / h
  }

  #Diferencia regresiva
  for (i in (length(x):2)) {
    fprimaReg <- (fx[i-1] - fx[i]) / (-h)

    if (!is.na(fprima[i])){
      if(fprimaReg != fprima[i]){
        aux <- fprima[i]
        fprima[i] <- glue::glue(aux, " (P)",
                                " o ",
                                fprimaReg, " (R)" )
      }
    } else{
      fprima[i] <- fprimaReg
    }

  }

  resultado <- data.frame(x, fx, fprima)

  return(resultado)
}
```

```
SplineNatural <- function(x, y){
 #browser()
  n <- length(x)

  # Paso 1
  h <- rep(NA, times = (n-1))
  for (i in 1:(n-1)) {
    h[i] <- x[i+1] - x[i]
  }; rm(i)
```

```r
# Paso 2
alfa <- rep(NA, times = (n-2))
for (i in 2:(n-1)) {
  alfa[i] <- (3/h[i]) * (y[i+1] - y[i]) - (3/h[i-1]) * (y[i] - y[i-1])
}

# Paso 3
mu <- rep(NA, times = n)
zeta <- rep(NA, times = n)
l <- rep(NA, times = n)

mu[1] <- 0
zeta[1] <- 0
l[1] <- 1


# Paso 4
for (i in 2:(n-1)) {
  l[i] <- 2 * (x[i+1] - x[i-1]) - h[i-1] * mu[i-1]
  mu[i] <- h[i]/l[i]
  zeta[i] <- (alfa[i] - h[i-1] * zeta[i-1])/l[i]
}

# Paso 5
l[n] <- 1
zeta[n] <- 0
c <- rep(NA, times = n)
c[n] <- 0

# Paso 6
b <- rep(NA, times = (n-1))
d <- rep(NA, times = (n-1))
for (j in (n-1):1) {
  c[j] <- zeta[j] - mu[j] * c[j+1]
  b[j] <- (y[j+1] - y[j]) / h[j] - h[j] * (c[j+1] + 2 * c[j])/3
  d[j] <- (c[j+1] - c[j]) / (3*h[j])
}

#Paso 7
resultados <- matrix(rep(NA, 4*(n-1)), nrow = (n-1), ncol = 4, byrow = F)
for (k in 1:(n-1)) {
  resultados[k, 1] <- y[k]
  resultados[k, 2] <- b[k]
  resultados[k, 3] <- c[k]
  resultados[k, 4] <- d[k]
}

print(resultados)

#Construyo el polinomio
polinomios <- rep(NA, times = nrow(resultados))
for (i in 1:nrow(resultados)) {
  polinomios[i] <- glue::glue(resultados[i,1])
```

```
    for(j in 2:ncol(resultados)){
      polinomios[i] <- polinomios[i] + glue::glue(" + ", resultados[i,j], " * (x - ", x[i], ")^", (j-1)
    }
  }

  return(polinomios)
}
```

```
SplineCondicionado <- function(x, y, fpo, fpn){
  #browser()
  n <- length(x)

  # Paso 1
  h <- rep(NA, times = (n-1))
  for (i in 1:(n-1)) {
    h[i] <- x[i+1] - x[i]
  }; rm(i)

  # Paso 2
  alfa <- rep(NA, times = n)
  alfa[1] <- 3 * (y[2] - y[1])/h[1] - 3 * fpo
  alfa[n] <- 3 * fpn - 3 * (y[n] - y[n-1]) / h[n-1]

  # Paso 3
  for (i in 2:(n-1)) {
    alfa[i] <- (3/h[i]) * (y[i+1] - y[i]) - (3/h[i-1]) * (y[i] - y[i-1])
  }; rm(i)

  # Paso 4
  mu <- rep(NA, times = n)
  zeta <- rep(NA, times = n)
  l <- rep(NA, times = n)

  l[1] <- 2 * h[1]
  mu[1] <- 0.5
  zeta[1] <- alfa[1]/l[1]



  # Paso 5
  for (i in 2:(n-1)) {
    l[i] <- 2 * (x[i+1] - x[i-1]) - h[i-1] * mu[i-1]
    mu[i] <- h[i]/l[i]
    zeta[i] <- (alfa[i] - h[i-1] * zeta[i-1])/l[i]
  }

  # Paso 6
  l[n] <- h[n-1] * (2 - mu[n-1])
  zeta[n] <- (alfa[n] - h[n-1] * zeta[n-1]) / l[n]
  c <- rep(NA, times = n)
  c[n] <- zeta[n]

  # Paso 7
  b <- rep(NA, times = (n-1))
```

```
  d <- rep(NA, times = (n-1))
  for (j in (n-1):1) {
    c[j] <- zeta[j] - mu[j] * c[j+1]
    b[j] <- (y[j+1] - y[j]) / h[j] - h[j] * (c[j+1] + 2 * c[j])/3
    d[j] <- (c[j+1] - c[j]) / (3*h[j])
  }

  #Paso 7
  resultados <- matrix(rep(NA, 4*(n-1)), nrow = (n-1), ncol = 4, byrow = F)
  for (k in 1:(n-1)) {
    resultados[k, 1] <- y[k]
    resultados[k, 2] <- b[k]
    resultados[k, 3] <- c[k]
    resultados[k, 4] <- d[k]
  }

  print(resultados)

  #Construyo el polinomio
  polinomios <- rep(NA, times = nrow(resultados))
  for (i in 1:nrow(resultados)) {
    polinomios[i] <- glue::glue(resultados[i,1])
    for(j in 2:ncol(resultados)){
      polinomios[i] <- polinomios[i] + glue::glue(" + ", resultados[i,j], " * (x - ", x[i], ")^", (j-1)
    }
  }

  return(polinomios)
}
```

Métodos

```
# Cargo el df
df <- data.frame("Tasa" = seq(from = 0, to = 0.1, by = 0.01), "Precio" = c(120, 114.56029, 109.42692, 1
```

A

Vuelvo a la definición de derivada:

$$\lim_{x \to \infty} f'(x) = \frac{f(x+h) - f(x)}{h}$$

```
derivadas <- DerivadaPorDefinicion(x = df$Tasa, fx = df$Precio)

print(paste("Derivada progresiva para P'(0):", derivadas$fprima[1]))
```

```
## [1] "Derivada progresiva para P'(0): -543.971"
```

```
print(paste("Derivada regresiva para P'(1):", derivadas$fprima[11]))
```

```
## [1] "Derivada regresiva para P'(1): -329.646"
```

B

2

Tasa	Precio
0.00	120.00000
0.01	114.56029
0.02	109.42692
0.03	104.57971
0.04	100.00000
0.05	95.67052
0.06	91.57527
0.07	87.69941
0.08	84.02916
0.09	80.55174
0.10	77.25528

DERIVACIÓN NUMÉRICA E INTERPOLACIÓN

Considere la tabla de tasas de interés (en tanto por uno) y Precio, suponiendo que el Precio es una función de la tasa: Precio = P(r).
a) Aproxime la derivada P'(0.00) y P'(0.10) usando el método que considere más preciso (justifique).
b) Estime P(0.095) mediante un Cubic Spline Natural.
c) Estime P(0.095) mediante un Cubic Spline Sujeto usando las derivadas estimadas en a).

[Escriba a continuación la respuesta, y cargue en el zip que entrega mediante campus el Script de R con la resolución.]
(15 puntos)

Figure 1: Consigna

```
trazadores <- SplineNatural(x = df$Tasa, y = df$Precio)
```

```
##             [,1]      [,2]      [,3]       [,4]
##  [1,] 120.00000 -550.5363    0.0000  65653.257
##  [2,] 114.56029 -530.8403 1969.5977 -21926.283
##  [3,] 109.42692 -498.0263 1311.8092   1871.876
##  [4,] 104.57971 -471.2285 1367.9655  -4221.221
##  [5,] 100.00000 -445.1356 1241.3289  -2256.993
##  [6,]  95.67052 -420.9861 1173.6191  -2750.806
##  [7,]  91.57527 -398.3390 1091.0949  -1579.784
##  [8,]  87.69941 -376.9910 1043.7014  -4710.058
##  [9,]  84.02916 -357.5300  902.3996   7640.015
## [10,]  80.55174 -337.1900 1131.6001 -37720.003
```

```
eval(parse(text = trazadores[10]), list(x = 0.095))
```

```
## [1] 78.88936
```

C

```
trazadores_condicionados <- SplineCondicionado(x = df$Tasa, y = df$Precio, fpo = (120 - 114.56029)/0.1,
```

```
##             [,1]      [,2]          [,3]         [,4]
##  [1,] 120.00000   54.3971 -104777.34927  4494053.927
##  [2,] 114.56029 -692.9337   30044.26854 -1208459.781
##  [3,] 109.42692 -454.5863   -6209.52489   319605.198
##  [4,] 104.57971 -482.8952    3378.63104   -88621.009
##  [5,] 100.00000 -441.9089     720.00075    17608.840
##  [6,]  95.67052 -422.2262    1248.26596     2185.649
##  [7,]  91.57527 -396.6052    1313.83542   -41191.435
##  [8,]  87.69941 -382.6859      78.09236   148800.092
##  [9,]  84.02916 -336.4841    4542.09513  -566788.934
## [10,]  80.55174 -415.6788  -12461.57290  2106485.645
```

```
eval(parse(text = trazadores_condicionados[10]), list(x = 0.095))
```

```
## [1] 78.42512
```