Teorema de Taylor

Juan Anagrita, HectorHernandez, Aldemar Ramirez

July 28,2019

Problema

Utilizando el teorema de Taylor hallar la aproximación de 0.5 e con cinco cifras significativas.

Solución

Lenguaje de programación: R

Funcion 1

La siguiente funcion calcula la n-esima derivada de una funcion

Parametros:

expr: funcion a derivar.

name: variables sobre la cual se va a derivar.

order: orden de la deriavda a calcular.

Retorna: expresion de la n-esima derivada de la funcion

```
DD <- function(expr, name, order = 1) {
  if(order < 1) stop("'order' must be >= 1")
  if(order == 1) D(expr, name)
  else DD(D(expr, name), name, order - 1)
}
```

Funcion 2(principal)

Funcion principal del Teorema de taylor

Parametros:

f: función sobre la cual se va a aplicar el Teorema de Taylor.

x0: valor sobre el cual se va a evaluar la función.

a: número alrededor del cual se evalua la función.

n: orden, ante los cual se va a derivar f.

```
taylorT = function(f, x0, a, n){ # f es tira
  # parse devuelve una expresión
  g = parse(text=f)
  # convertir en función
  fx = function(x){eval(g[[1]])}
  # almacenar los sumandos
  smds = rep(NA, length=n+1)

  for(k in 1:n){
    g. = DD(g,"x", k)
```

```
    fp = function(x) eval(g.)
    smds[k]=1/factorial(k)*(x0-a)^k *fp(a)
  }

  smds[n+1] = fx(a)
  sum(smds)
}
```

retorna:

$$p(x) = \sum_{k=0}^{n} (f^{(k)} * (x_0) * (x - x_0))/k!$$

Funcion 3: error

Función que permite una aproximación al error al calcular una expresión hasta el orden n con el teorema de Taylor.

Parametros:

n: orden ante el cual se calcula el teorema de Taylos.

maxLocal: maximo local de la función en el rango a,b.

```
error = function(n,maxLocal,a,b){
  return(b^(n+1)/factorial(n+1) * maxLocal)
}
```

retorna:

$$R_n(x) = x^{(n+1)} * f^{(n+1)}(E_x)/(x+1)!$$

Implementación

Asignación de valores inciales

Los diferentes valores de los parametros para calcular e^(0.5).

Se calculara a través del Polinomio de Taylor con grados en el rango 1:6, para comparar el error, y el valor de la expresión. El rango es de -1 a 1.

```
f <- expression(exp(1)^(x))
x0 <- 0.5
a <- 0

maxLocal <- exp(1)
a <- -1
b <- 1
grados <-c(1:10)
```

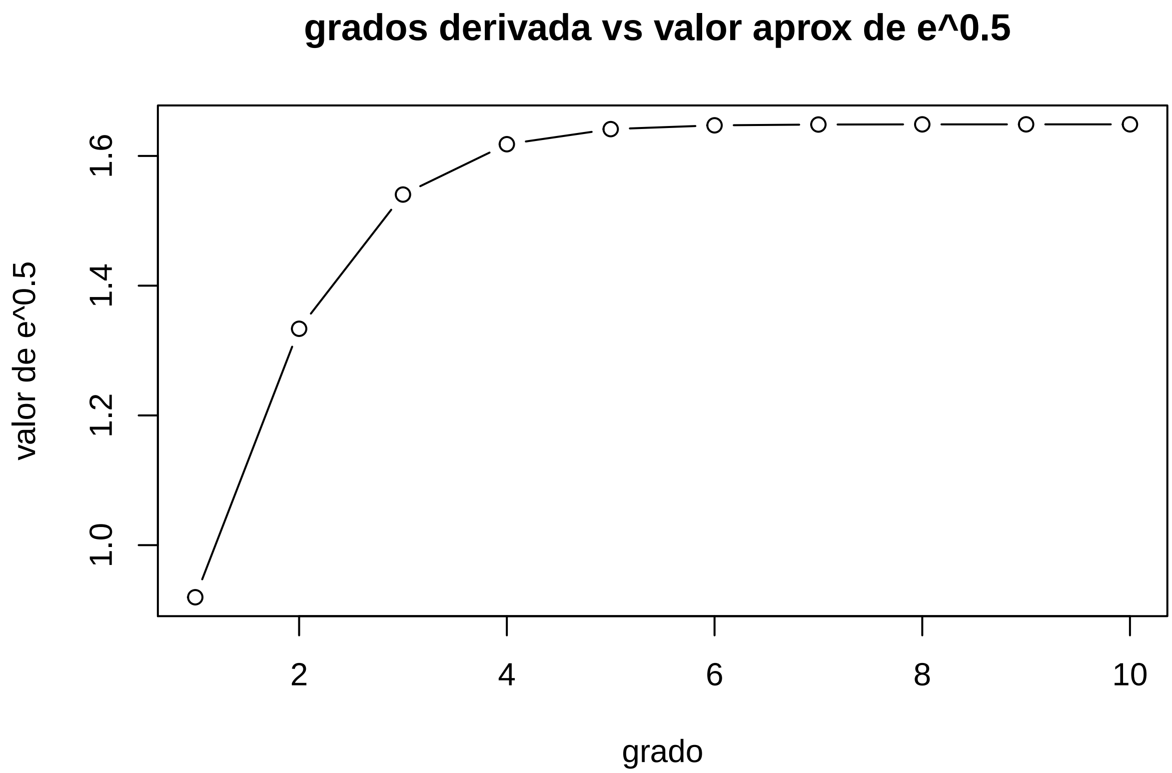
Tabla de resultados

```r
resultados <- c()
errores <- c()

for( i in grados){
  resultados = c(resultados,taylorT(f, x0, a, i))
  errores = c(errores,error(i,maxLocal,a,b))
}

tabla <- data.frame(
  "grado" = grados,
  "Aprox. e^0.5" = resultados,
  "Error" = errores
)

print(tabla)
```

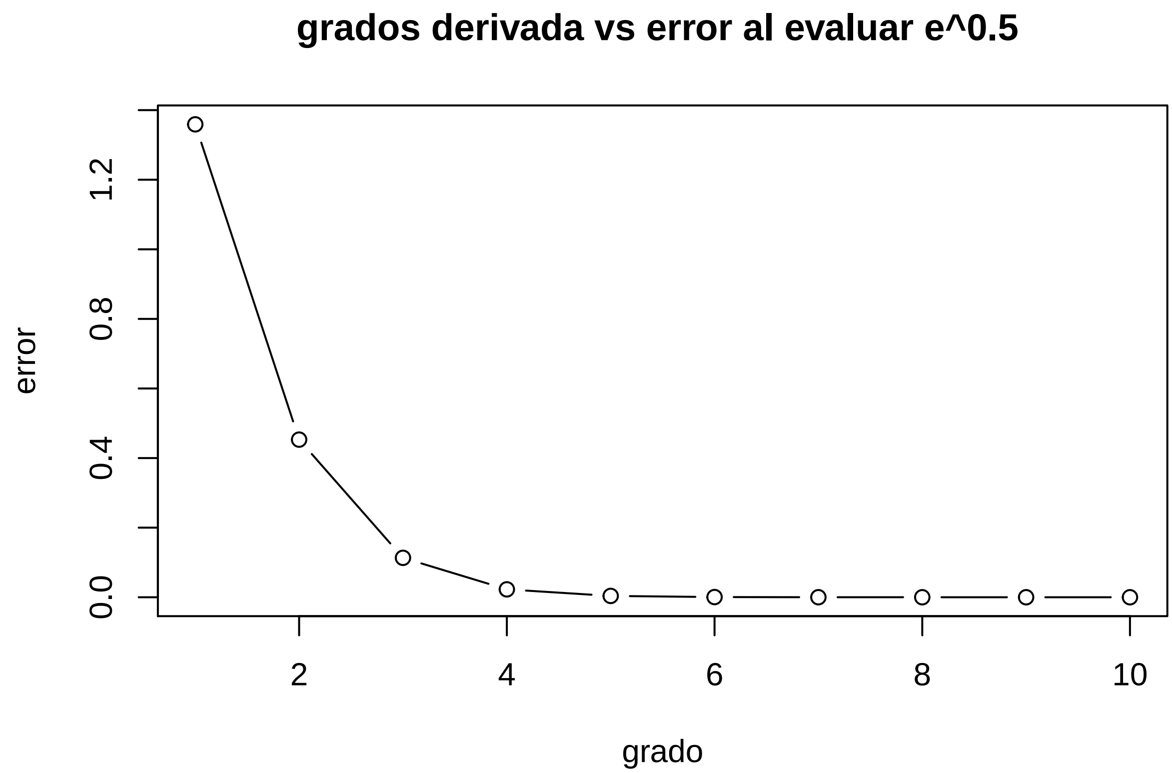
```
##    grado Aprox..e.0.5        Error
## 1      1    0.9196986 1.359141e+00
## 2      2    1.3335630 4.530470e-01
## 3      3    1.5404952 1.132617e-01
## 4      4    1.6180947 2.265235e-02
## 5      5    1.6413746 3.775391e-03
## 6      6    1.6471946 5.393416e-04
## 7      7    1.6484417 6.741770e-05
## 8      8    1.6486755 7.490856e-06
## 9      9    1.6487145 7.490856e-07
## 10    10    1.6487204 6.809869e-08
```

Gráfica

```r
plot(x = grados, y = resultados, type = "b", xlab="grado", ylab="valor de e^0.5",
     main = "grados derivada vs valor aprox de e^0.5 ")
```




```
plot(x = grados, y = errores, type = "b", xlab="grado", ylab="error",
     main = "grados derivada vs error al evaluar e^0.5 ")
```



Como se ve en las gráficas entre más terminos sean usados en el teorema de Taylor para evaluar la expresión mayor va a ser la exactitud. El problema podría estar en la complejidad de calcular las primeras n derivadas de una función.

Resultados

Como se muestra en tabla anterior para calcular e^0.5 hasta la quinta cifra significativa de manera correcta, el polinomio de taylor hay que evaluarlor la novena derivada.

```
options(digits = 5)
cat("e^0.5 =(aprox) ",taylorT(f, x0, a, 9),"\n")
```

```
## e^0.5 =(aprox)  1.6487
```

```
cat("Error < ",error(9,maxLocal,a,b),"\n")
```

```
## Error <  7.4909e-07
```