Informe Fundamentos Derivadas

Marta Divassón Carribero

Actividad 1: Encontrar la derivada de 3*x^2-2*x+4~x y calcular el valor de la derivada en x=0

Primero, cargamos las librerías:
library(mosaicCalc)
library(mosaic)

```
##Ejercicios de Cálculo 1


#Exercise 1
##Encontrar la derivada de una función
#Primero cargar librerías
library(mosaicCalc)
## Loading required package: mosaicCore
##
## Attaching package: 'mosaicCalc'
## The following object is masked from 'package:stats':
##
##     D
library(mosaic)
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:mosaicCore':
##
##     count, tally
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
## Loading required package: lattice
## Loading required package: ggformula
## Loading required package: ggplot2
## Loading required package: ggstance
##
## Attaching package: 'ggstance'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh
##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")
## Loading required package: mosaicData
## Loading required package: Matrix
##
## The 'mosaic' package masks several functions from core packages in
order to add
## additional features.  The original behavior of these functions shou
ld not be affected by this.
##
## Note: If you use the Matrix package, be sure to load it BEFORE load
ing mosaic.
##
## Attaching package: 'mosaic'
## The following object is masked from 'package:Matrix':
##
##     mean
## The following object is masked from 'package:ggplot2':
##
##     stat
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median,
##     prop.test, quantile, sd, t.test, var
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

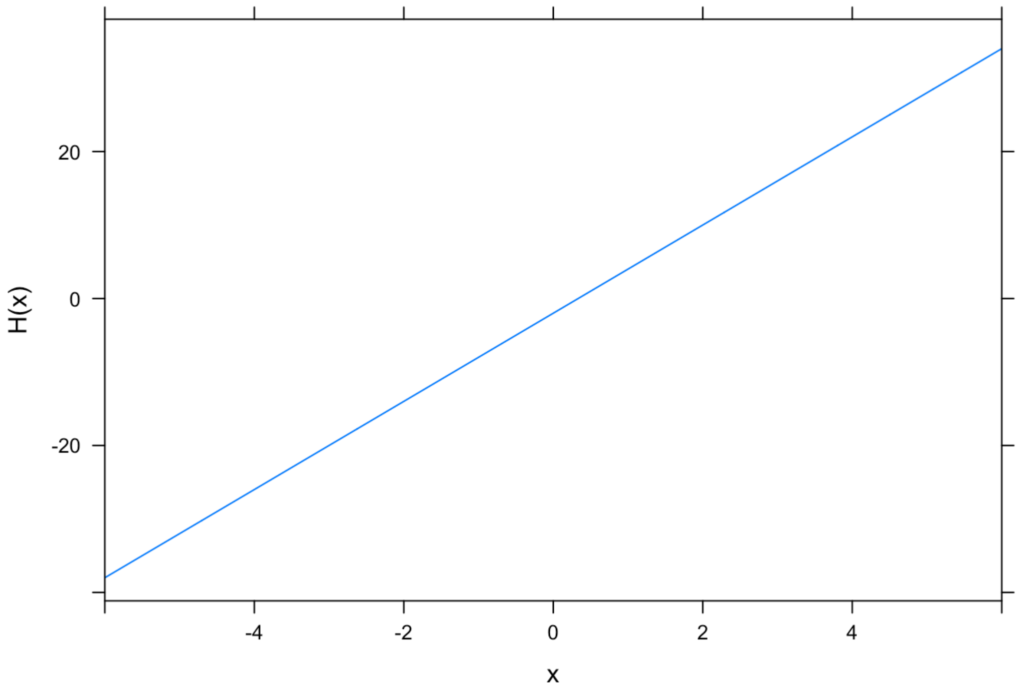
```
H=mosaicCalc::D(3*x^2-2*x+4~x)
H
```

```
## function (x)
```

```
## 3 * (2 * x) - 2
##Valor de la derivada en x=0
H(0)
## [1] -2
##La gráfica de la función derivada y tiene un positive sloping line
plotFun(H, x.lim=range(-6,6))
```

Después, ejecutamos la función: H=mosaicCalc::D(3*x^2-2*x+4~x) y se obtiene que la derivada es en function (x)=**3 * (2 * x) – 2.**
Su valor cuando x=0, se calcula ejecutando H(0), y se obtiene que: **H(0) [1] -2**

La gráfica de la función derivada es upward sloping y se obtiene ejecutando: **plotFun(H, x.lim=range(-6,6))**



<u>Actividad 2:</u> Encontrar la derivada de 5*exp(.2*x)~x y calcular el valor de la derivada en x=0

```
#Exercise 2
I=mosaicCalc::D(5*exp(.2*x)~x)
I
## function (x)
## 5 * (exp(0.2 * x) * 0.2)
#Valor de la derivada en x=0
```

```
I(0)
## [1] 1
#Gráfico de la funcion derivada
g=makeFun(5*exp(.2*x)~x)
g(x=1)
## [1] 6.107014
g(x=5)
## [1] 13.59141
plotFun(g(x)~x,x.lim=range(-5,5))
```

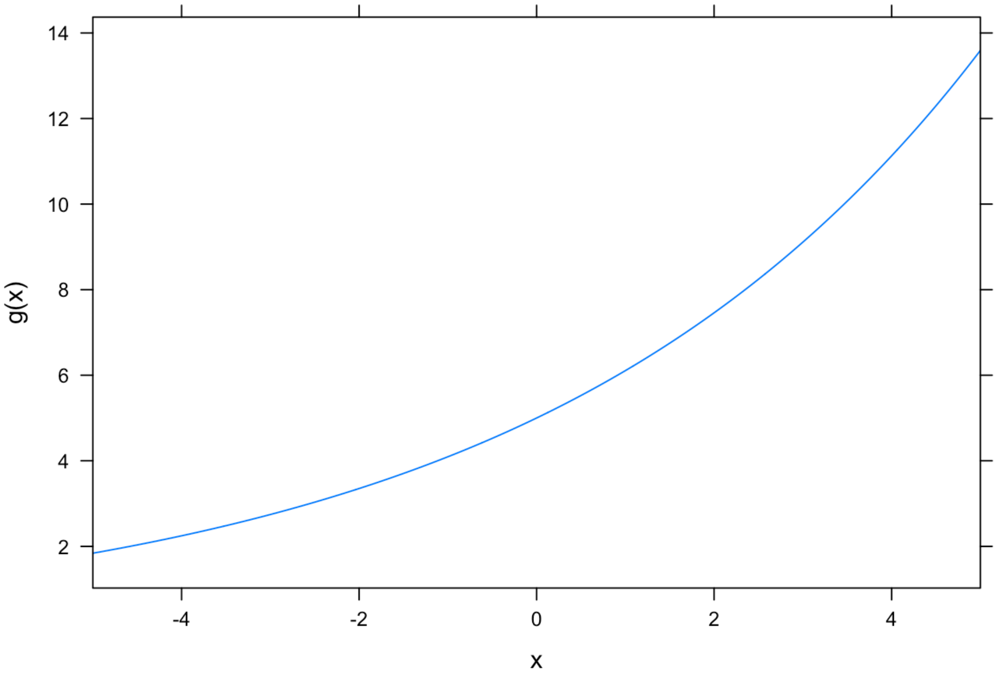
Ejecutamos la función: I=mosaicCalc::D(5*exp(.2*x)~x) y se obtiene que la derivada es en función(x) **5 * (exp(0.2 * x) * 0.2).**

Su valor en x=0, es D(0); [1] **1.**

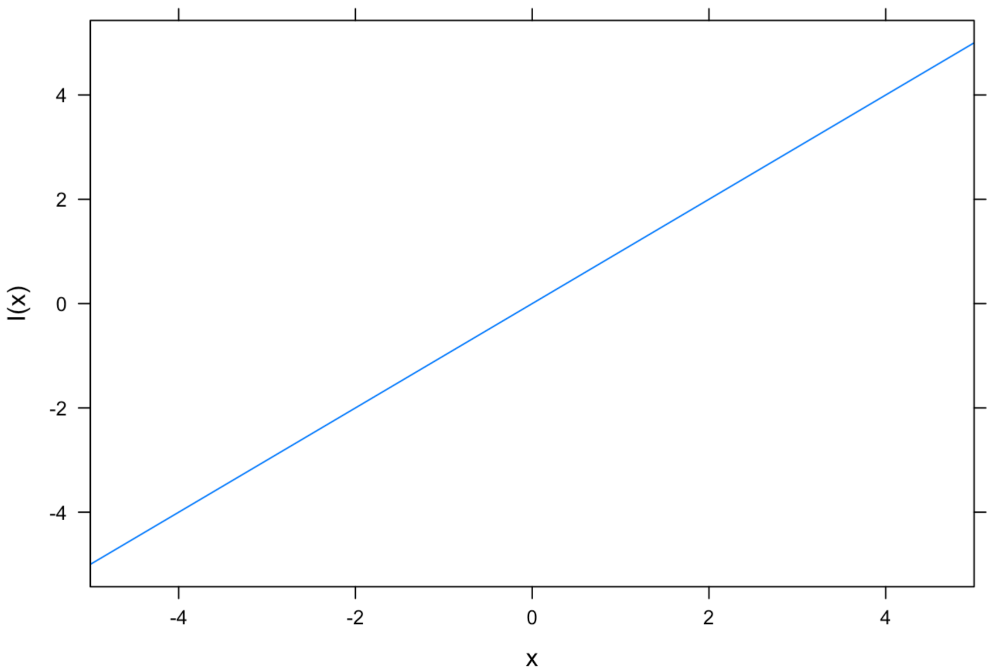
Luego, para obtener valores en la función inicial se ejecuta:
g=makeFun(5*exp(.2*x)~x)
g(x=1)
[1] 6.107014
g(x=5)
[1] 13.59141

plotFun(g(x)~x,x.lim=range(-5,5)) y esta función para representarla gráficamente.

Con respecto a la gráfica, la de la derivada incrementa exponencialmente. Y la de la función original, si le damos a x una serie de valores, obtenemos una gráfica que crece exponencialmente más rápido.




```
plotFun(I,x.lim =range(-5,5))
```

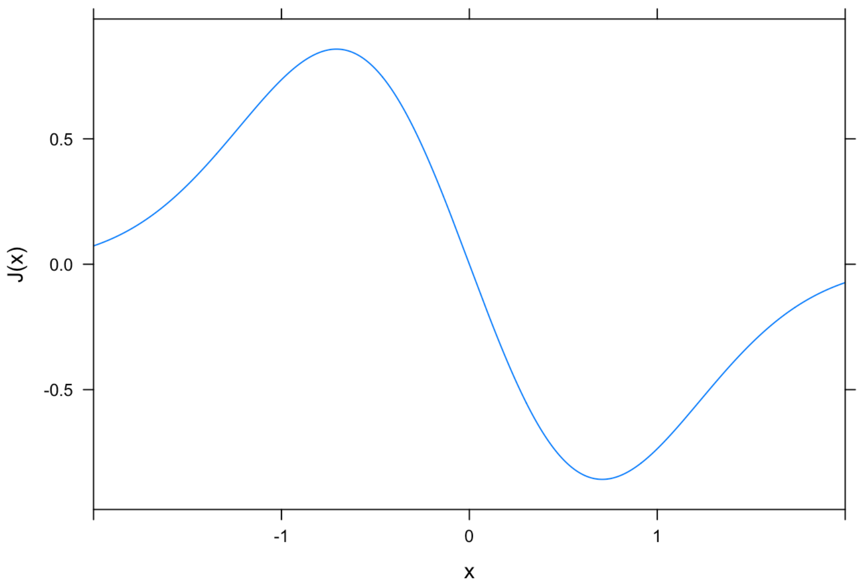


Actividad 3: Calcular la derivada de exp(-x^2)~x y representar en el rango del -2 al 2.

```
#Exercise 3
##Calcular la derivada y represnetar en rango del -2 al 2
J=mosaicCalc::D(exp(-x^2)~x)
J
## function (x)
## -(exp(-x^2) * (2 * x))
plotFun(J,x.lim=range(-2,2))
```

Ejecutamos la función: J=mosaicCalc::D(exp(-x^2)~x) y se obtiene que la derivada es en función de x, **-(exp(-x^2) * (2 * x)).**

Se ejecuta la siguiente para obtener un gráfico: plotFun(J,x.lim=range(-2,2)); y se obtiene que es un wave positivo seguido de uno negativo.




```
##Es un positive wave followed by a negative wave
```

Actividad 4: Calcular el valor de la siguiente derivada: D(fred^2~ginger) y representarlo gráficamente.
Se ejecuta mediante: d=mosaicCalc::D(fred^2~ginger) y el resultado es **function (ginger, fred) 0.** Esto quiere decir que es 0 en toda la función y al representarla gráficamente **plotFun(d,x.lim=range(0,10)),** da error.

```
#Exercise 4
#Valor de la siguiente funcion: D(fred^2~ginger); 0 everywhere
d=mosaicCalc::D(fred^2~ginger)
d
## function (ginger, fred)
## 0
##no se representa graficamente ya que es 0 en todala función
```

Actividad 5: Utilizar la función cos(2*t) ~t&t&t para encontrar la tercera derivada y la cuarta derivada.

Tercera derivada: **K=mosaicCalc::D(cos(2*t)~t&t&t&t)** con resultado:
function (t)
cos(2 * t) * 2 * 2 * 2 * 2 o lo que es lo mismo, **8sin(2t)**

Cuarta derivada: **L=mosaicCalc::D(cos(2*t)~t&t&t&t)** con resultado:
function (t)
cos(2 * t) * 2 * 2 * 2 * 2 o **16cos(2t)**

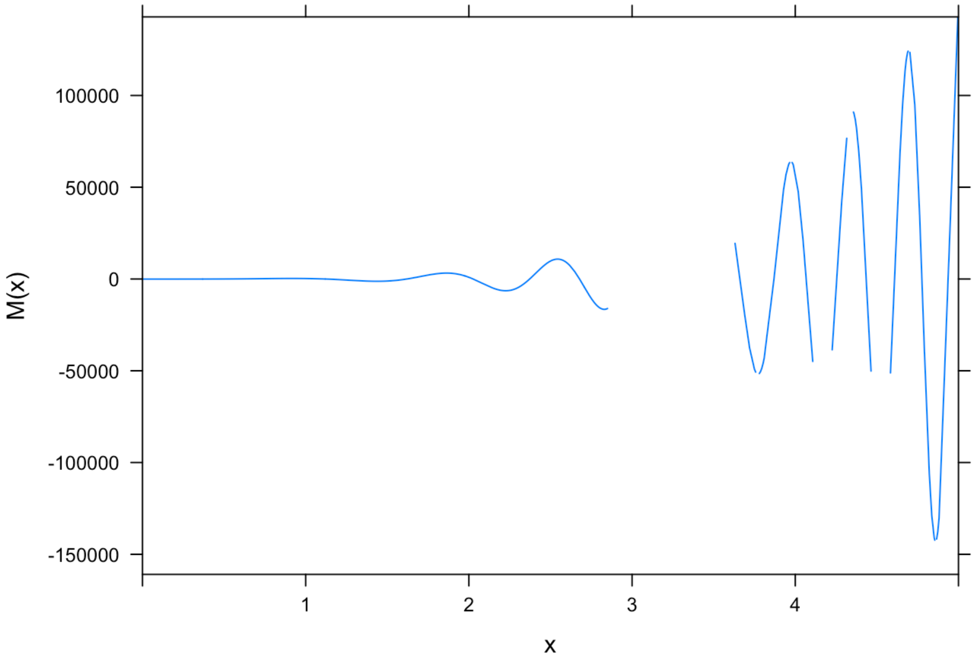
```
#Exercise 5
#Tercera derivada de cos(2*t) ~t&t&t = 8sin(2t)
K=mosaicCalc::D(cos(2*t)~t&t&t)
K
## function (t)
## sin(2 * t) * 2 * 2 * 2
##4 derivada =16cos(2t)
L=mosaicCalc::D(cos(2*t)~t&t&t&t)
L
## function (t)
## cos(2 * t) * 2 * 2 * 2 * 2
```

Actividad 6: Computar la gráfica de la cuarta derivada de cos(2*t^2) ~t con rango del 0 al 5

```
##Exercise 6

#Cuarta derivada de cost(2*t^2) ~t con rango del 0 al 5

M=mosaicCalc::D(cos(2*t^2)~t&t&t&t)

M

## function (t)

## -((cos(2 * t^2) * (2 * 2) - sin(2 * t^2) * (2 * (2 * t)) * (2 *

##     (2 * t))) * (2 * 2) - (sin(2 * t^2) * (2 * (2 * t)) * (2 *

##     2) + ((cos(2 * t^2) * (2 * (2 * t)) * (2 * (2 * t)) + sin(2 *

##     t^2) * (2 * 2)) * (2 * (2 * t)) + sin(2 * t^2) * (2 * (2 *

##     t)) * (2 * 2))) * (2 * (2 * t)) + (cos(2 * t^2) * (2 * 2) -

##     sin(2 * t^2) * (2 * (2 * t)) * (2 * (2 * t))) * (2 * 2) +

##     (cos(2 * t^2) * (2 * 2) - sin(2 * t^2) * (2 * (2 * t)) *

##         (2 * (2 * t))) * (2 * 2))

plotFun(M,x.lim=range(0,5))
```

M=mosaicCalc::D(cos(2*t^2)~t&t&t&t) con resultado:
function (t)
-((cos(2 * t^2) * (2 * 2) - sin(2 * t^2) * (2 * (2 * t)) * (2 *
(2 * t))) * (2 * 2) - (sin(2 * t^2) * (2 * (2 * t)) * (2 *
2) + ((cos(2 * t^2) * (2 * (2 * t)) * (2 * (2 * t)) + sin(2 *
t^2) * (2 * 2)) * (2 * (2 * t)) + sin(2 * t^2) * (2 * (2 *
t)) * (2 * 2))) * (2 * (2 * t)) + (cos(2 * t^2) * (2 * 2) -
sin(2 * t^2) * (2 * (2 * t)) * (2 * (2 * t))) * (2 * 2) +
(cos(2 * t^2) * (2 * 2) - sin(2 * t^2) * (2 * (2 * t)) *
(2 * (2 * t))) * (2 * 2))

Aparecen sin, cos, multiplicación, elevado al cuadrado y suma
Y al representarla gráficamente con la función plotFun(M,x.lim=range(0,5)) obtenemos lo siguiente:



En la que la amplitud del coseno aumenta y su periodo disminuye

```
#Aparecen sin, cos, multiplicacion, elevado al cuadrado y suma; en la
que la amplitud del coseno aumenta y su periodo disminuye
```

Actividad 7: Considerar la función x*sin(y) y calcular lo siguiente:
pxy=D(x*sin(y))~x&y
pxy=D(x*sin(y))~y&x

```
#Exercise 7

#Considerar la funcioón:x*sin(y) y calcular lo siguiente:

pxy=D(x*sin(y))~x&y

pxy=D(x*sin(y))~y&x

#Derivada parcial con respecto de x

N=mosaicCalc::D(x*sin(y)~x)

N

## function (x, y)

## sin(y)

#Derivada parcial con respecto de y

O=mosaicCalc::D(x*sin(y)~y)

O

## function (y, x)

## x * cos(y)

#Segunda derivada parcial con respecto de x

P=mosaicCalc::D(x*sin(y)~x&x)
```

```
P
## function (x, y)
## 0
#Segunda derivada parcial con respecto de y
Q=mosaicCalc::D(x*sin(y)~y&y)
Q
## function (y, x)
## -(x * sin(y))
#Mezcla de las dos
pxy=mosaicCalc::D(x*sin(y)~x+y)
pxy
## function (x, y)
## cos(y)
pyx=mosaicCalc::D(x*sin(y)~y+x)
pyx
## function (y, x)
## cos(y)
#Las parciales con respecto de x e y no son identicas
#Las segundas derivadas parciales con respecto de x e y no son identic
as
#Las dos parciales combinadas son identicas
```

Primero, calcular la derivada parcial con respecto de x:
D=mosaicCalc::D(x*sin(y)~x) con resultado: **function (x, y)**
sin(y)

Luego, la derivada parcial con respecto de y
D=mosaicCalc::D(x*sin(y)~y) con resultado: **function (y, x = 1)**
x * cos(y)

Segunda derivada parcial con respecto de x
D=mosaicCalc::D(x*sin(y)~x&x) con resultado: **function (x, y)**
0

Segunda derivada parcial con respecto de y
D=mosaicCalc::D(x*sin(y)~y&y) con resultado: **function (y, x = 1)**
-(x * sin(y))

Y finalmente, la mezcla de las dos:
pxy=mosaicCalc::D(x*sin(y)~x+y)
pxy

solución: function (x, y)
cos(y)
pyx=mosaicCalc::D(x*sin(y)~y+x)
pyx
function (y, x)
cos(y)
Con esto se deduce que las parciales con respecto de x e y no son idénticas ; Las segundas derivadas parciales con respecto de x e y no son idénticas y las dos parciales combinadas son identicas.