Breve Instructivo de Operación

wxMaxima

(versión Windows)

Ing. Agr. Victor Prieto

Depto. Biometría, Estadística y Computación
Facultad de Agronomía – UdelaR

Índice de contenido

	Página
1) Instalación del programa	3
2) Entorno de trabajo y ejecución de órdenes	4
3) Definición de una función	5
4) Derivación e Integración	6
5) Gráfico de funciones	10
6) Funciones estadísticas	12
7) Álgebra de matrices	15
Anexo:	
Funciones de más de una variable independiente	20

1) Instalación del programa

La forma efectiva de instalar el programa es mediante el respectivo programa instalador disponible en el sitio web *Sourceforge*, repositorio de diversas herramientas de software libre. En el enlace http://sourceforge.net/projects/maxima/files/Maxima-Windows/ se encuentran las últimas versiones del programa, se recomienda la descarga de la versión más actual.

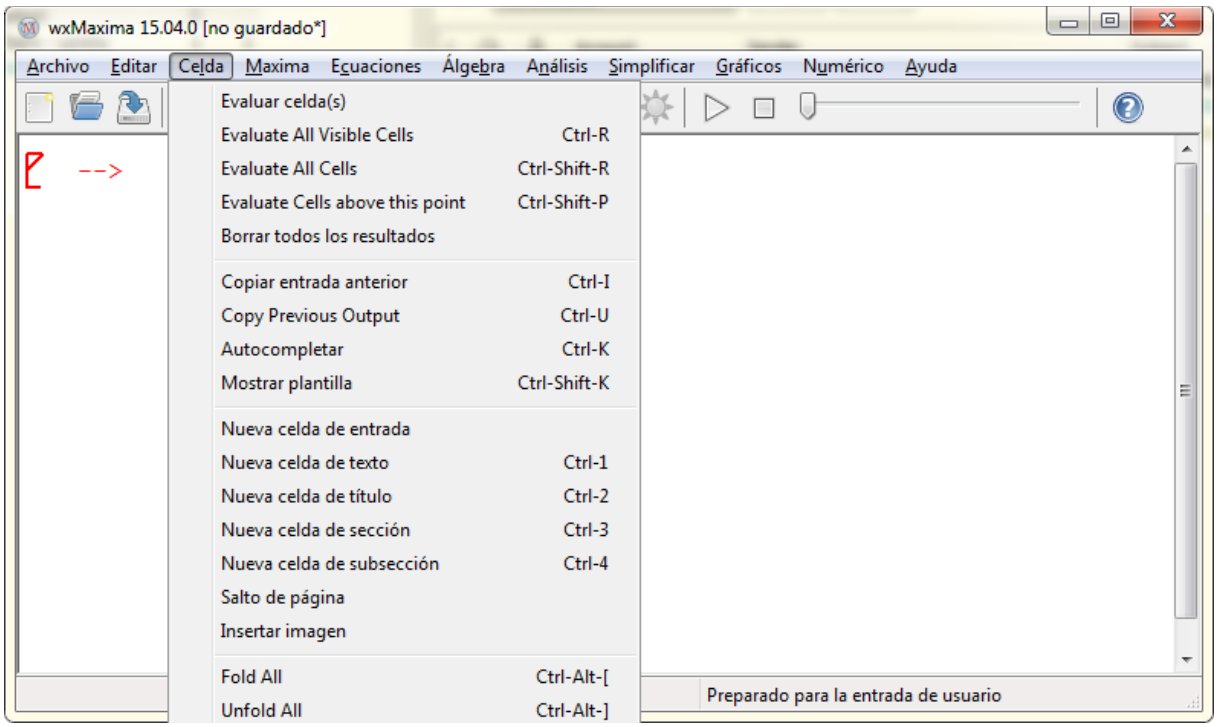
La descarga del programa instalador tiene por nombre "maxima-sbcl-x.xx.x.exe, donde los términos "x.xx.x" indica la versión y sub-versión del mismo. A vía de ejemplo, éste breve instructivo se eleboró con la versión 5.31.3.

Una vez ejecutado el programa instalador, y luego de completar un asistente que lo guía Ud. debe aceptar la licencia GNU de uso de software libre, lo que finaliza el proceso.

Dispondrá entonces de un nuevo ícono en el menú de programas de Windows, y si lo prefiere en el Escritorio, un menú llamado "Maxima 5.xx.x" y la aplicación "wxMaxima" con el ícono siguiente:

2) Entorno de trabajo

El programa *Maxima*, que es una aplicación mediante comandos, posee una interface de operación amigable llamada *wxMaxima*. La misma nos provee de un menú principal y una barra de herramientas de uso frecuente:



Como toda aplicación de *Windows*, las tareas de operación aparecen agrupadas en el menú principal según la tarea específica que realiza. Por ejemplo, en el menú "Celda" encontrará las principales operaciones para la ejecución de comandos.

A modo de ejemplo, si quisiéramos evaluar una expresión matemática debemos primero crear una celda de entrada yendo al menú *Celda* > *Nueva celda de entrada*, o simplemente teclear enter.

```
-->  (3+5)*7
```

Luego de ingresar la expresión deseada, evaluar la expresión yendo al menú *Celda* > *Evaluar celda(s)*. Otra forma alternativa de evaluar una celda y obtener el mismo resultado es teclear la secuencia Ctrl+enter.

```
(%i1)  (3+5)*7;
(%o1)  56
```

Como vemos, el resultado de la expresión aparece seguidamente a la operación que lo origina. El símbolo "%i1" significa que es la primer sentencia de entrada o input ("i"), y "%o1" significa que es la primer salida ó output ("o"). Toda sentencia en Máxima debe finalizar con un punto y coma (";").♦

3) Definición de funciones

La forma de definir la relación funcional entre dos variables numéricas (x, f(x)) es mediante el símbolo ":=" que relacione la variable dependiente f(x) con una una o más variables independientes (x_i). Por ejemplo, el modelo lineal particular puede ingresarse de la siguiente manera:

```
(%i1)  f(x):= 2+3*x;
(%o1)  f(x):=2+3 x
```

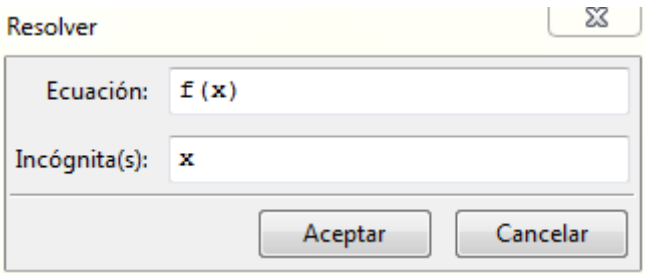
De esta manera queda definida la función [f(x)] como el polinomio de primer grado que tiene como término independiente el valor 2 y pendiente igual a 3.

Una vez definida la función podemos operar con ella mediante su nombre. Por ejemplo para obtener el valor funcional para los valores de x=0 y x=10 debemos ingresar:

```
(%i2)  f(0);f(10);
(%o2)  2
(%o3)  32
```

Obtenemos el resultado deseado de f(0) = 2 y f(10) = 32

Para hallar los ceros de la función, es decir, el valor de x que hace f(x)=0 entonces recurrimos al menú *Ecuaciones* > *Resolver* e indicamos la ecuación (en éste caso la función f), y la incógnita o variable x.



Luego de aceptar obtenemos la raíz de la función f que es x=-2/3. Notamos que la opción *Resolver* del menú crea una celda y ejecuta el código siguiente:

```
(%i4)  solve([f(x)], [x]);
(%o4)  [x=-2/3]
```

Para definir la función derivada de f(x) o f'(x) utilizamos el menú *Análisis* > *Derivar.* Indicamos la función a derivar, la variable por la cual derivamos y el número de veces que hacemos la derivación (indicamos 1 para la derivada primera). Luego, con el comando %i6 definiremos como g(x) a la función derivada primera de f(x):

```
(%i5)  diff(f(x),x,1);
(%o5)  3
(%i6)  define(g(x), diff(f(x),x,1));
(%o6)  g(x):=3
```

♦

4) Derivación e Integración

Derivación

Para realizar el estudio analítico y representación gráfica de una función es indispensable estudiar la función derivada primera y segunda.

Ejemplo 1: Definir la función $f(x) = 2 - 3x + x^2$, estudie ceros de f(x), f´(x), f´´(x) , hallar los extremos de la función f(x).

```
(%i1) f(x):=2-3*x+x^2$
      functions;
      fundef(f);
(%o2) [f(x)]
(%o3) f(x):=2-3 x+x^2
```

La celda %i1 consta de tres sentencias, y su evaluación nos da dos salidas. Esto es debido a que la primera termina con el símbolo "$" que impide el despliegue en pantalla del resultado.

En el caso de la sentencia functions nos informa cuales son las funciones hasta ahora definidas y la sentencia fundef (cuyo argumento es el nombre de una función) devuelve la expresión de dicha función.

```
(%i4) 'diff(f(x),x,1);
      define(f\'(x), diff(f(x),x,1));
      functions;
(%o4) d/dx (x^2-3 x+2)
(%o5) f'(x):=2 x-3
(%o6) [f(x),f'(x)]
```

En la celda %i4 evaluamos tres sentencias y obtenemos tres salidas: la primer sentencia, que es el comando diff() es para obtener la derivada de una función, que al estar precedida por el símbolo "'" impide su evaluación, por lo que solamente devuelve la expresión simbólica. En la siguiente sentencia, donde se define f´(x) a partir de f(x) despliega la ecuación que toma la derivada primera. Por último la sentencia functions que nos informa que existen ahora dos funciones definidas.

```
(%i7) r:solve(f(x))$
      r;
      r[1];
      f(r[1]);
(%o8) [x=1,x=2]
(%o9) x=1
(%o10) x^2-3 x+2=0
```

En ésta celda el comando solve() devuelve las raíces de la función f(x) pero esta vez guardamos el resultado en un objeto llamado "r". La forma de hacerlo es con el símbolo ":" de asignación.

El objeto r consta de dos elementos, que son las raíces del polinomio. La forma de referirse a ellos es especificando su orden: en el ejemplo el primer elemento lo obtengo con la sentencia r[1]; de esa forma puedo trabajar con los objetos o resultados utilizándolos como argumento en otras sentencias.

En el caso de las funciones polinómicas existe el comando allroots() siendo el argumento de la sentencia una función o expresión de tipo polinómica. En el ejemplo siguiente obtenemos los ceros de la función f(x) y f'(x):

```
(%i11) allroots(f(x));
       allroots(f\'(x));
(%o11) [x=1.0,x=2.0]
 (%o12) [x=1.5]
```

f(x)	++	0	--	0	++
x		1		2	

El resultado de la celda %i11 indica que existen dos raíces reales en x=1 y x=2 (ver diagrama de signo de f).

Es el caso de la derivada primera, ésta presenta una raíz en x=1.5. Es decir que hemos encontrado un punto crítico de la función f(x) ya que su derivada primera cambia de signo.

f'(x)	--	0	++
x		1.5	

En el diagrama, la derivada pasa de tener signo negativo (f(x) decrece) a positivo (f(x) crece).

Estudiando la derivada segunda de la función podemos obtener información sobre su concavidad y el tipo de punto crítico presente (máximo, mínimo o punto de inflexión).

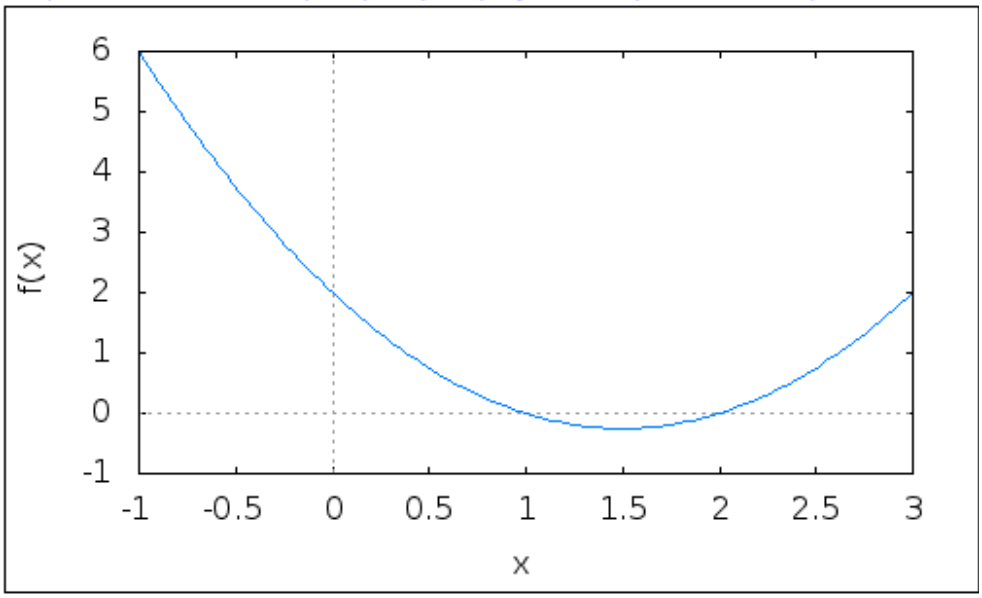
Debemos primero definir en el programa una función a la que derivamos dos veces y luego hallamos sus ceros:

```
(%i13) define(f\'\'(x), diff(f(x),x,2));
       allroots(f\'\'(x));
(%o13) f''(x):=2
 (%o14) [ ]
```

Observamos que la derivada segunda f''(x) es una constante positiva, por lo que el resultado de la función allroots() es un conjunto vacío ("[ ]"). Esto implica concavidad positiva de f(x) o crecimiento con incrementos crecientes. Por lo tanto, el punto crítico en x=1.5 es un mínimo.

Por último, verificamos estas conclusiones realizando un gráfico o plot. El siguiente comando grafica f (en ordenada) en función de x (en abscisa), estableciendo el dominio de la función para valores de x en el intervalo [-1, 3]:

```
(%i15) wxplot2d([f(x)],[x,-1,3],[ylabel,"f(x)"]);
```



(%t15)

(%o15)

Más adelante veremos las diferentes opciones de graficación de *Maxima*.

Integración

Cálculo de área bajo la curva

Ejemplo 2: Definir la función f(x) = 2 – 3x + x^2, hallar una primitiva de la función f(x) y calcular el área bajo la curva de f(x) encerrada entre x =2 y x =3.

En primera instancia definimos f(x) y F(x) con el comando define(...). De esa forma obtenemos una primitiva de la función que nos interesa:

```
(%i1) define(f(x), 2-3*x+x^2);
      define(F(x), integrate(f(x),x));
```

(%o1) $f(x):=x^2-3x+2$

(%o2) $F(x):=\frac{x^3}{3}-\frac{3x^2}{2}+2x$

Vemos como la forma de obtener la integral o primitiva de una expresión es con el comando integrate() que tiene como argumentos: *la expresión a integrar* - f(x) en nuestro ejemplo - y *la variable integradora* -la variable independiente x -.

Sabemos que F(x) es una de las infinitas primitivas de f(x), solemos simbolizar ese hecho haciendo F(x) + c , donde c puede tomar cualquier valor constante. Verificamos ese hecho haciendo una derivación de la primitiva para obtener la función f(x).

$$\boxed{f(x)} \rightarrow \text{integrate}(...) \rightarrow \boxed{F(x) + c} \rightarrow \text{diff}(...) \rightarrow \boxed{f(x)}$$

En el caso del área encerrada entre la función y las abscisas x=2 y x=3 debemos aplicar el teorema de la integral definida, o *regla de Barrow*, que es la diferencia entre F(x=3) y F(x=2).

Dado que F(3) = 3/2 y F(2) = 2/3, la diferencia es entonces de 5/6. Por lo tanto, el área bajo la curva deseada es igual al valor 5/6 ó 0,8333. El mismo resultado se obtiene con el comando integrate solo que debemos especificar los límites de la integración:

```
(%i3) integrate(f(x),x,2,3);
(%o3) 5/6
```

Función integrate() con integral definida entre abscisas x=2 y x=3.

Si queremos obtener el resultado en formato decimal, en la siguiente sentencia asignamos el valor de la integral definida a un objeto llamado “area” y agregamos el parámetro numer para conmutar la salida a tipo numérico.

```
(%i4) area:integrate(f(x),x,2,3)$
      area;
      area,numer;
(%o5) 5/6
(%o6) .8333333333333334
```

♦

5) Gráfico de funciones

Las propiedades de graficación de *Maxima* son variadas y se encuentran desarrolladas en el manual de ayuda del programa. En éste apartado veremos únicamente la forma simple de graficar la relación funcional entre dos variables de forma tal que y=f(x).

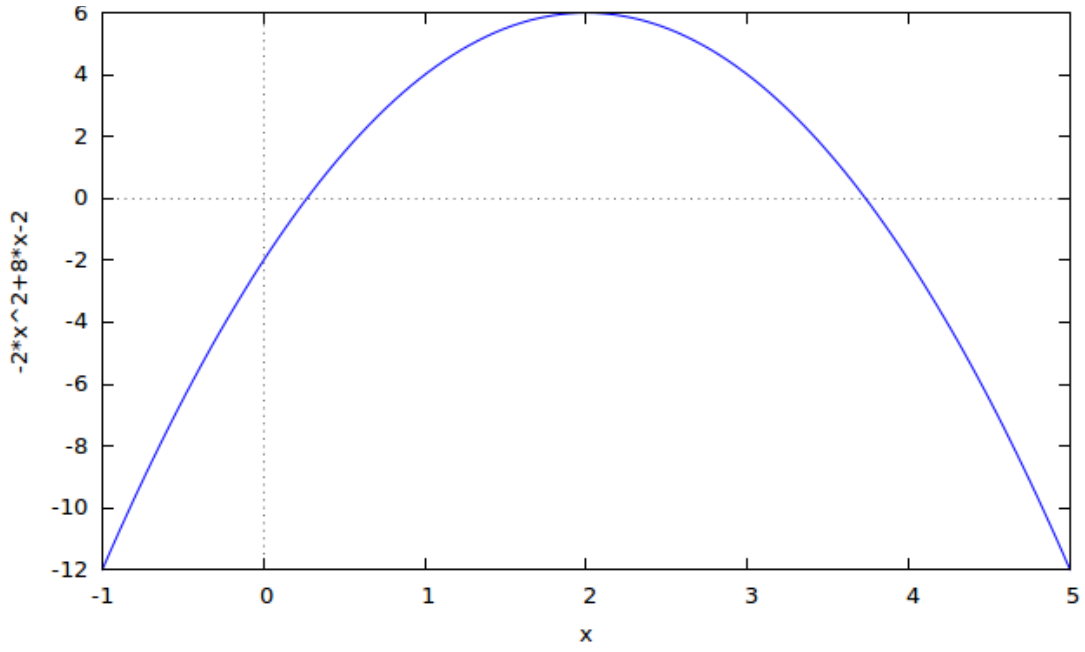
Ejemplo 3: Definir la función f(x) = -2 + 8x - 2 x^2 y grafique en un par de ejes cartesianos xy para el intervalo de x en [-1, 5]. Defina la función derivada primera de f(x) y grafique en el mismo par de ejes.

```
(%i1) define(f(x), -2+8*x-2*x^2);
      define(f\'(x), diff(f(x),x));
(%o1) f(x):=-2*x^2+8*x-2
(%o2) f'(x):=8-4*x
```

Definimos f(x) y f'(x)

Para el caso del gráfico en 2 dimensiones, es necesario ejecutar el comando plot2d() y especificar la variable en la ordenada y el dominio de x en la abscisa:

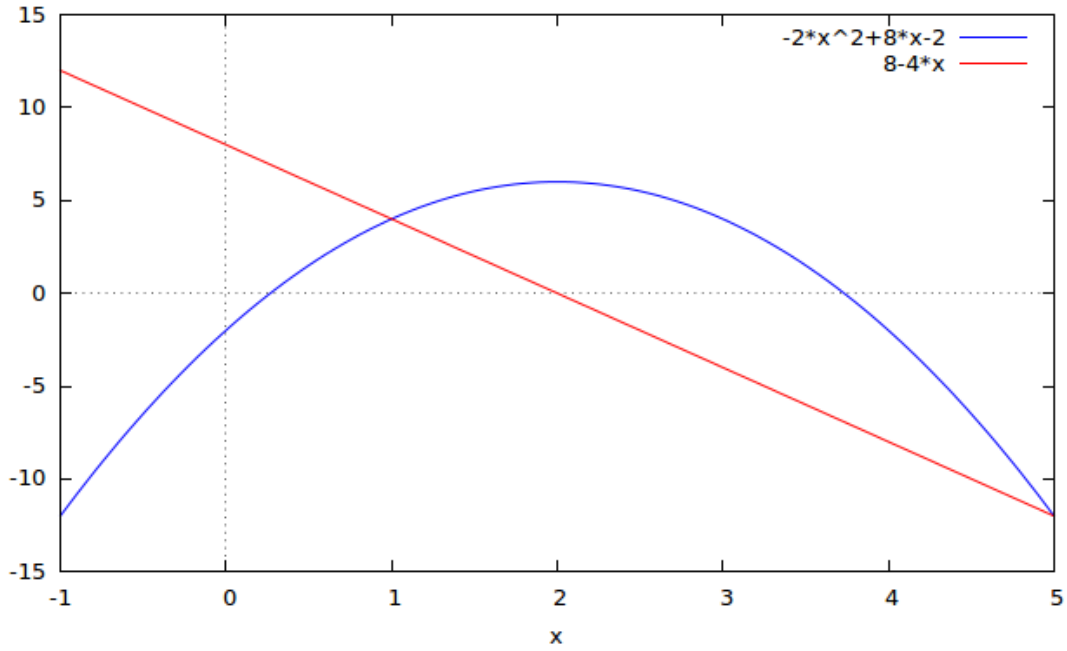
```
(%i3) plot2d(
      f(x),
      [x,-1,5]);
(%o3)
```



En la medida que deseamos graficar más de una variable en la ordenada, podemos definir el conjunto de funciones como una lista de expresiones a graficar.

El código siguiente grafica entonces f(x) y f'(x) en un mismo par de ejes:

```
(%i4) plot2d(
      [f(x),f\'(x)],
      [x,-1,5]);
(%o4)
```



Para mejorar la salida gráfica, pueden agregarse opciones tales como etiquetas al eje x y al eje y, colores de línea diferentes y el texto de la leyenda.

Ejecutamos el siguiente código para el despliegue del gráfico deseado y damos por terminada la resolución del ejemplo:

```
(%i5) plot2d(
      [f(x),f\'(x)],
      [x,-1,5],
      [xlabel,"Variable x"],
      [ylabel,"Variable f(x)"],
      [color, black, red],
      [legend, "f(x)","f\'(x)"]);
(%o5)
```

♦

6) Funciones estadísticas

El programa *Maxima* es un software enfocado al análisis matemático, aunque posee componentes auxiliares para resolver otro tipo de problemas. Tal es el caso de las funciones estadísticas. El presente capítulo propone realizar la descripción estadística de una variable numérica, a través del módulo descriptive de *Maxima*.

Inicialmente debemos cargar el módulo de estadística descriptiva en memoria con el comando load(descriptive). Conviene además ejecutar la orden numer para conmutar la salida de resultados numérica:

```
--> load (descriptive);
    numer: true;
```

Posteriormente ingresamos a través de comando una serie de datos de interés:

```
--> peso:
    [378,391,389,398,401,
     396,384,410,399,419,
     423,402,405,399,420,
     402,397,400,410,389]$
```

Como ejemplo uilizaremos el registro del peso vivo (en kg) de 20 vacunos dedicados a la producción lechera. Asignamos a esa lista el nombre de "peso".

En caso de desconocer el número de datos que consta una lista, podemos hallar ese valor con la función length().

```
(%i4) npeso  : length(peso);
(%o4) 20
```

Para variables cuantitativas como el peso animal (en kg, nivel de medición de proporción) es posible describir adecuadamente el conjunto de datos mediante indicadores de posición, dispersión y asimetría.

Tabla de funciones para la descripción estadística mediante indicadores

Posición		**Dispersión**	
Nombre	función	Nombre	función
Mínimo	smin()	Rango	range()
Máximo	smax()	I.Intercuart (IQ)	qrange()
Media	mean()	Varianza	var()
Mediana (Q2)	median()	Desvío estándar	std()
Cuartil 1 (Q1)	quantile(.., .25)	Coef. Variación	cv()
Cuartil 3 (Q3)	quantile(.., .75)		
Asimetría			
Coef. Asimetría de Pearson	pearson_skewness()		

Éste listado de funciones no es exaustivo: el programa provee de más indicadores útiles por lo que se recomienda recurrir al manual del programa para otras opciones.

Luego de ejecutar éstas funciones estadísticas y asignar el resultado a un objeto obtenemos la salida ordenada de indicadores:

```
(%i5) min   : smin(peso);
      max   : smax(peso);
      media: mean(peso);
      Me    : median(peso);
      Q1    : quantile(peso,.25);
      Q2    : quantile(peso,.50);
      Q3    : quantile(peso,.75);
(%o5) 378
 (%o6) 423
 (%o7) 400.6
 (%o8) 399.5
 (%o9) 394.75
(%o10) 399.5
(%o11) 406.25
```

```
(%i12) rango: range(peso);
       IQ   : qrange(peso);
       S2   : var(peso);
       S    : std(peso);
       cvar : cv(peso);
(%o12) 45
 (%o13) 11.5
 (%o14) 130.54
 (%o15) 11.42541027709727
 (%o16) .02852074457587937
(%i17) sesgo: pearson_skewness(peso);
(%o17) .2815165543975712
```

Así vemos que los 20 registros de ésta población en estudio, tienen un peso medio de 400.6 kg y un peso mediano de 399.5. El primer cuartil (deja antes de sí al 25% de los individuos) tiene un valor de 394.5 y el tercer cuartil (deja antes de sí el 75%) vale 406.25. El rango de la variable es de 45 kg (ya que el mín= 378 y máx= 423) con una varianza de 130.54 kg^2 y desviación típica (o estándar) de 11.43 kg. La variabilidad relativa, esto es su coeficiente de variabilidad, es del 2.85% (cv·100). La distribución de valores de peso posee una asimetría positiva (sesgo= +0.28).

Agrupamiento de la variable

En el caso que tengamos la necesidad de realizar un cuadro o un gráfico de una variable cuantitativa, debemos en primer término agrupar los valores en clases mutuamente excluyentes y así obtener la tabla de frecuencias.
Para ello existen dos formas en *Maxima*: definir el número de clases deseada o los límites de clases deseada.

El siguiente comando define para los datos del ejemplo, un agrupamiento de la variable en 6 clases homogéneas:

```
(%i18) continuous_freq (peso, 6);
(%o18) [[378,385.5,393.0,400.5,408.0,415.5,423.0],[2,3,6,4,2,3]]
```

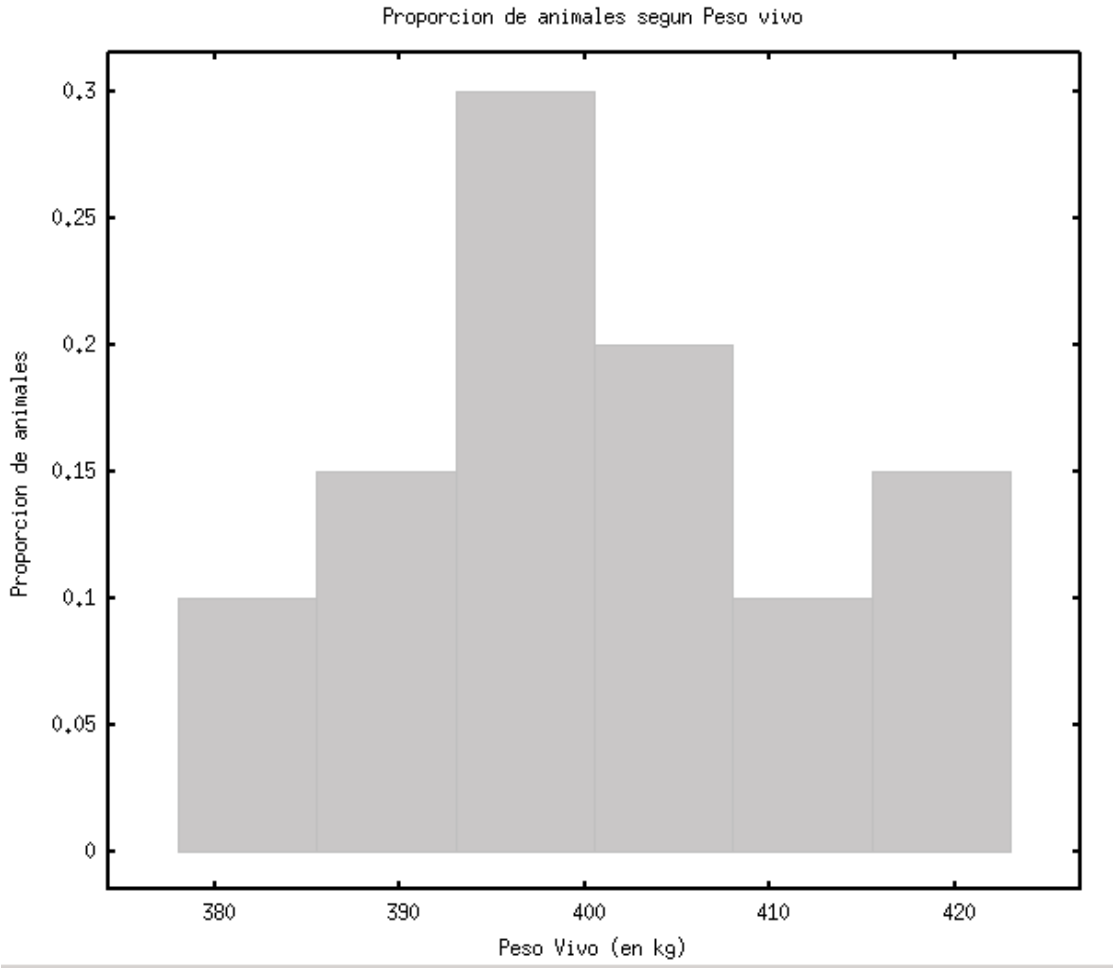
El resultado de la ejecución del comando es el siguiente: un vector de 7 elementos que son los límites de clases (límite inferior y superior de las 6 clases) y un vector de 6 elementos que son sus respectivas frecuencias absolutas.

Descripción mediante gráficos

Para una descripción gráfica fiel de éstos datos es necesario realizar un histograma de frecuencias de los datos. Para ello utilizamos la función histograma() a la que le definimos como argumentos: la variable, el número de clases, el tipo de frecuencia a graficar (absoluta o relativa) y una serie de características de visualización. Como ejemplo ejecutamos el siguiente código:

```
(%i19) histogram(peso,
                 nclasses     = 6,
                 title        = "Proporcion de animales segun Peso vivo",
                 xlabel       = "Peso Vivo (en kg)",
                 ylabel       = "Proporcion de animales",
                 frequency    = relative,
                 fill_color   = grey,
                 fill_density = 0.6)$
```

Para obtener como resultado el siguiente histograma:



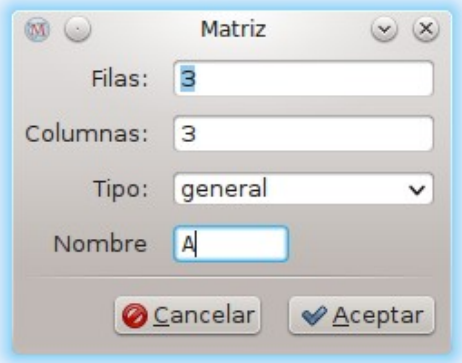
♦

7) Álgebra de matrices

Para ingresar una matriz se puede recurrir al menú *Álgebra* > *Introducir matriz*, indicar las dimensiones de la matriz y su nombre. Luego llenar los valores correspondientes a cada elemento de la matriz. A modo de ejemplo ingresaremos la siguiente matriz al entorno de trabajo de *Maxima*:

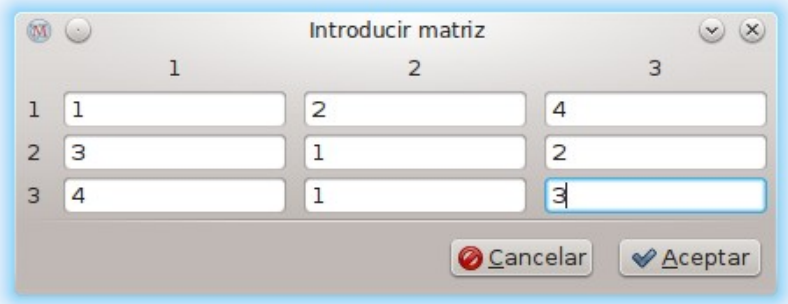
$$A=\begin{pmatrix}1 & 2 & 4\\ 3 & 1 & 2\\ 4 & 1 & 3\end{pmatrix}$$

Paso 1: menú *Álgebra* > *Introducir matriz*



Definir dimensión (filas y columnas de la matriz.

Paso 2:



Ingresar los elementos de la matriz.

Código para definir otra matriz B de igual dimensión:

```
--> B: matrix(
       [7,3,1],
       [2,3,5],
       [8,1,6]
      );
```

Vemos entonces que la función matrix() incluye las filas de la matriz separadas por comas, y dentro de cada fila disponemos entre paréntesis recto los elementos de las columnas. Por supuesto, entre las filas debe haber el mismo número de elementos para conformar una matriz.

De la misma manera podemos definir un vector fila o un vector columna introduciendo el siguiente comando:

```
--> x: matrix(
      [1],
      [4],
      [2]
    );
```

vector columna

```
--> y: matrix(
      [2,1,7]
    );
```

vector fila

Una vez definidas las matrices podemos hacer todas las operaciones escalares y matriciales básicas, tal como suma, resta y multiplicación.

```
(%i5) A+B;
```

$$(\%o5)\begin{bmatrix} 8 & 5 & 5 \\ 5 & 4 & 7 \\ 12 & 2 & 9 \end{bmatrix}$$

```
(%i6) A-B;
```

$$(\%o6)\begin{bmatrix} -6 & -1 & 3 \\ 1 & -2 & -3 \\ -4 & 0 & -3 \end{bmatrix}$$

Si intentamos realizar una operación entre matrices de dimensión inapropiada, un mensaje de error nos alertará del problema. Por ejemplo, realizar la suma entre un vector columna y un vector fila:

```
(%i7) x+y;
fullmap: arguments must have same formal structure.
 -- an error. To debug this try: debugmode(true);
```

Para la multiplicación entre un escalar y una matriz, así como entre matrices (si ésta es posible) se debe utilizar el operador "." o producto matricial no-conmutativo. Seguidamente se presentan algunos ejemplos:

```
(%i8) 2 .x;
```

$$(\%o8)\begin{bmatrix} 2 \\ 8 \\ 4 \end{bmatrix}$$

```
(%i9) A.B;
```

$$(\%o9)\begin{bmatrix} 43 & 13 & 35 \\ 39 & 14 & 20 \\ 54 & 18 & 27 \end{bmatrix}$$

```
(%i10) A.x;
```

$$(\%o10)\begin{bmatrix} 17 \\ 11 \\ 14 \end{bmatrix}$$

Tener en cuenta que al ser producto de matrices, no conmutativo, puede que el producto entre matrices no sea posible ya que las dimensiones no son conformables. Tal es el caso de pre-multiplicar un vector columna por una matriz cuadrada:

```
(%i11) x.A;
MULTIPLYMATRICES: attempt to multiply nonconformable matrices.
 -- an error. To debug this try: debugmode(true);
```

Determinantes

Para el cálculo del determinante de una matriz, es necesario utilizar la función determinant(). En el caso de una matriz cuadrada, de dimensión 2, sabemos que el determinante es la diferencia entre productos cruzados. En máxima lo implementamos de la siguiente manera:

```
(%i12) M: matrix([a,b],[c,d]);
       determinant( M );
(%o12) [ a  b ]
       [ c  d ]
(%o13) a d-b c
```

Determinante de una matriz genérica, de tamaño 2.

Para matices de mayor tamaño, Máxima utiliza un método similar al de eliminación gaussiana, que permite el cálculo de determinante de matrices de cualquier tamaño, siempre que sea una matriz cuadrada. Para la matriz A de nuestro ejemplo, el determinante tiene un valor de

```
(%i14) determinant(A);
(%o14) -5
```

Se recomienda repasar el cálculo para matrices de tamaño 3 y 4 por el método de adjuntos. Éste consiste como primer paso en el cálculo de los elementos cofactores de la matriz, con éstos luego multiplicar y sumar para una columna o fila cualesquiera de la matriz para obtener el valor del determinante:

Matriz de cofactores: es la transpuesta de la matriz adjunta

```
(%i15) Adj: adjoint(A);
       [  1   -2   0 ]
       [                ]
(%o15) [ -1  -13  10 ]
       [                ]
       [ -1    7   -5 ]
```

Llamamos entonces como matriz Adj a la resultante de la matriz adjunta de A

```
(%i16) transpose(Adj);
       [  1   -1   -1 ]
       [                ]
(%o16) [ -2  -13   7  ]
       [                ]
       [  0   10   -5 ]
```

Al transponer la matriz adjunta obtenemos la matriz de cofactores

Podemos verificar entonces que al multiplicar y sumar los elementos de cualquier fila o columna de la matriz A con su correspondiente cofactor obtenemos siempre el -5, el valor del determinante de A.

Inversión de matrices

Conceptualmente, cuando dos matrices cuadradas A y B satisfacen la igualdad

$$AB = BA = I$$

donde la matriz *I* es una matriz de identidad, decimos que éstas matrices están en relación inversa. Decimos entonces que $A = B^{-1}$ y $B = A^{-1}$ siempre que la inversa exista.

A modo de ejemplo, hallaremos la matriz C^{-1} con Máxima:

```
(%i17) C: matrix([1,6],[5,2]);
(%o17) [ 1  6 ]
       [ 5  2 ]
```

Definimos la matriz de ejemplo C

```
(%i22) invert(C);
(%o22) [ -1/14   3/14 ]
       [  5/28  -1/28 ]
```

Hallamos la matriz C^{-1}

```
(%i23) C.invert(C);
(%o23) [ 1  0 ]
       [ 0  1 ]
```

Verificamos el resultado
$C \cdot C^{-1} = I$

Para el cálculo de la matriz inversa se puede utilizar la fórmula

$$A^{-1} = \text{adj}(A) / \det(A) \qquad [1]$$

por lo cual es necesario que el valor de det (A) sea diferente de 0.

Si al mismo comando utilizado para la inversión de la matriz C, le agregamos un modificador de salida podemos desplegar el resultado de la forma en [1].

```
(%i24) invert(C), detout;
                [ 2  -6 ]
(%o24) -        [ -5  1 ]
       ----------------------
                  28
```

La función detout modifica la salida de forma que el determinante aparezca como un factor.

det(C) = -28

Resolución de sistema de ecuaciones

La utilidad de la matriz inversa se pone de manifiesto cuando pretendemos hallar la solución a un sistema de ecuaciones, donde la misma se resuelveusualmente mediante álgebra de matrices.

Supongamos el siguiente sistema de ecuaciones:

$$\begin{aligned} 3x_1+2x_2&=-1 \\ x_1-x_2&=1 \end{aligned} \quad \text{ó} \quad \begin{pmatrix} 3 & 2 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}=\begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \text{de forma matricial}$$

De ésta forma, al plantear el sistema de la forma matricial Ax=y obtendremos la solución al sistema mediante $A^{-1}y = x$.

```
(%i25) A: matrix([3,2],[1,-1]);
       y: matrix([-1],[1]);
```

$$(\%o25)\ \begin{bmatrix} 3 & 2 \\ 1 & -1 \end{bmatrix}$$

$$(\%o26)\ \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

La solución al sistema es por lo tanto:

```
(%i27) x: invert(A).y;
```

$$(\%o27)\ \begin{bmatrix} \frac{1}{5} \\ -\frac{4}{5} \end{bmatrix}$$

Como resultado obtenemos entonces el valor de $x_1 = 0.2$ y $x_2 = -0.8$

♦

ANEXO:

Funciones de más de una variable independiente

La operativa del programa para el análisis de una función con más de una variable independiente es la misma. Cabe aclarar que se deben definir explícitamente las variables independientes y operar con las mismas.

Por ejemplo, la función f(x, y) = 5 +2x − $3y^2$ + 2xy vincula la variable dependiente con las variables independientes x, y a través de una relación funcional polinómica.

```
(%i1) define( f(x,y), 5+2*x-3*y^2+2*x*y);
(%o1) f(x,y):=-3 y^2+2 x y+2 x+5
```

```
(%i2) f(x,0);
      f(0,y);
      f(0,0);
(%o2) 2 x+5
(%o3) 5-3 y^2
(%o4) 5
```

Una vez definida la función podemos operar. Si sólo nos interesa la función considerando la variable x, hacemos constante la variable y (ej: y=0), quedando definida como función de una sola variable independiente. Si hacemos constante tanto x como y (ej: x = y = 0) entonces obtenemos la ordenada para esa coordenada (x,y).

Para el estudio de las derivadas parciales recordamos el comando diff() de las secciones anteriores, pero ahora debemos indicar por cual variable independiente estamos derivando la función:

```
(%i5) diff(f(x,y),x);
      diff(f(x,y),y);
(%o5) 2 y+2
(%o6) 2 x-6 y
```

Comando *diff()* para hallar derivadas parciales primeras de la función f(x, y)

Puede ser útil obtener las derivadas parciales hallando la matriz jacobiana. Recordemos que ésta matriz está compuesta en sus elementos por las derivadas parciales de primer orden de una (o varias funciones).

```
(%i8) jacobian([f(x,y)],[x,y]);
(%o8) [2 y+2  2 x-6 y]
```

En nuestro ejemplo, si tenemos una única variable dependiente y dos variables independientes (x, y) entonces es un vector fila de dimensión 2.

Se sugiere dirigir la salida del comando *jacobian()* a un objeto (que llamaremos "J") que será creada por defecto como una matriz de 1 fila y 2 columnas. Luego podemos referenciar los elementos de esa matriz dependiendo de cuál derivada

parcial estemos buscando. Lo vemos en el comando siguiente:

```
(%i10) J:jacobian([f(x,y)],[x,y])$
       J[1,1];
       J[1,2];
(%o11) 2 y+2
 (%o12) 2 x-6 y
```

El elemento $J_{1,1}$ es entonces $\frac{\partial f(x,y)}{\partial x}$

Siguiendo con éste análisis, utilizaremos el comando *diff()* para hallar las derivadas parciales de segundo orden.

Derivadas parciales segundas:

```
(%i7) diff(f(x,y),x,2);
      diff(f(x,y),y,2);
      diff(f(x,y),x,1,y,1);
      diff(f(x,y),y,1,x,1);
(%o7) 0
 (%o8) -6
 (%o9) 2
(%o10) 2
```

En estos comandos vemos el argumento que indica por cual variable derivamos 2 veces. Notar los dos últimos comandos, como se deriva de forma cruzada.

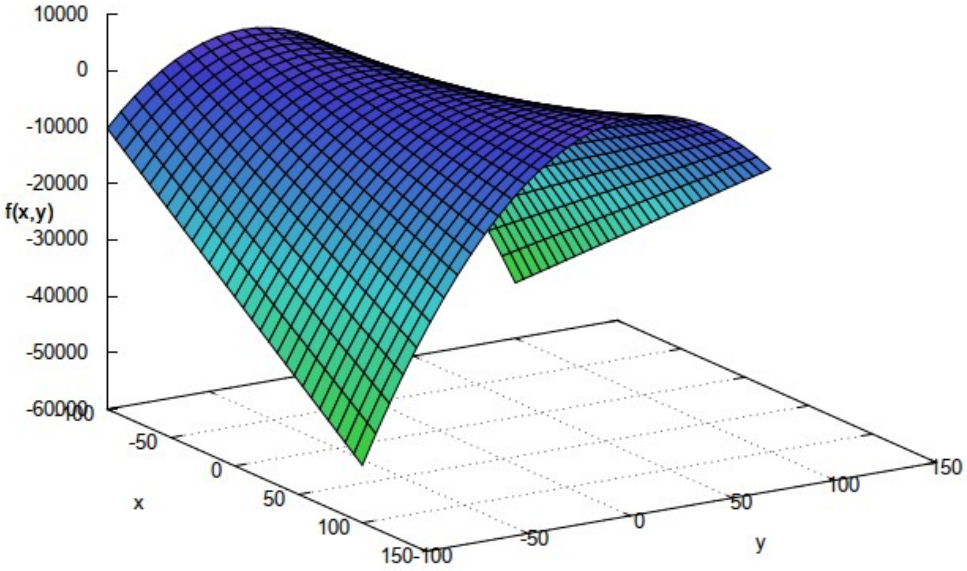
Como sabemos, las derivadas segundas participan como criterio en la búsqueda de extremos de una función. Las mismas pueden condensarse en una matriz de derivadas de segundo orden, llamada matriz hessiana. En nuestro ejemplo, de una función de dos variables independientes se obtiene una matriz hessiana de 2 filas y dos columnas:

```
(%i14) hessian(f(x,y),[x,y]);
(%o14) matrix([0,2],[2,-6])
```

Reconozca las derivadas de *x* e *y* en la diagonal, y derivadas cruzadas en el resto de los elementos de la matriz

Por último, es necesario considerar las posibilidades que nos brinda Máxima para la graficación de funciones de varias variables. Es posible obtener el gráfico de superficie (es decir, en 3D) si se trata de una función de dos variables independientes. En el siguiente comando obtenemos el gráfico:

```
(%i16) plot3d(
       f(x,y),
       [x,-100,100],
       [y,-100,100],
       [xlabel,"x"],
       [ylabel,"y"],
       [zlabel, "f(x,y)"],
       [legend, " Gráfico en 3D de f(x,y) "])$
```



Vale decir que *Maxima* posee diversidad de opciones de graficación. Se recomienda el comando *draw3d()* de la librería draw que permite no solo el mismo gráfico de superficie que *plot3d()* sino además gráficos de contorno, ideales para el estudio de curvas a nivel de la función. Se remite al lector explorarlas en el manual de usuario.

♦