Práctica 2:
Álgebra Lineal con *Mathematica*

Grado en Administración y Dirección de Empresas

José Manuel Cascón Barbero (casbar@usal.es)
María Dolores García Sanz (dgarcia@usal.es)
Bernardo Ramón García-Bernalt Alonso (bgarcia@usal.es)
María Aurora Manrique García (amg@usal.es)
Gustavo Santos García (santos@usal.es)

1. Introducción

Mathematica es una gran herramienta para visualizar y realizar cálculos muy potentes. En particular, la capacidad de este software para las operaciones con matrices, la resolución de sistemas de ecuaciones, el cálculo de valores y vectores propios, etc., permite una docencia de la asignatura de Álgebra Lineal más ágil y centrada en los conceptos, además de atraer la atención de nuestros estudiantes, para los que el Álgebra pasa a ser una asignatura mucho más visual e interactiva. Podemos también poner en práctica todo lo aprendido en clase con un número de datos elevado o con matrices de dimensiones mayores sin que ello ocasione problemas de cálculo. A trabajar con los comandos dedicados a estas cuestiones dedicamos este capítulo. Comenzamos aprendiendo a introducir vectores y matrices en la sección 2 para continuar en la sección 3 con sus operaciones básicas. Dedicamos la sección 4 a la resolución de sistemas de ecuaciones, la sección 5 a aplicaciones lineales y la sección 6 a diagonalización de endomorfismos. El estudio de las formas cuadráticas ocupa la sección 7 y terminamos con un ejemplo ilustrativo del tema en la sección 8.

2. Vectores y Matrices

En *Mathematica* los vectores y matrices se escriben como listas. Eso quiere decir que para su definición se deben emplear las llaves: {}:

Vector: lista

In[1]:=
```
u = {1, 2, 4}
```
Out[1]=
```
{1, 2, 4}
```

In[2]:=
```
v = {2, 3, 5}
```
Out[2]=
```
{2, 3, 5}
```

Matriz: Es una lista de sublistas (vectores). Cada sublista contiene los elementos de cada fila.

In[3]:=
```
A = {{0, 1, 0}, {0, 0, 1}, {2, -5, 4}}
```
Out[3]=
```
{{0, 1, 0}, {0, 0, 1}, {2, -5, 4}}
```

In[4]:=
```
B = {{1, 2}, {3, 0}, {3, 4}}
```
Out[4]=
```
{{1, 2}, {3, 0}, {3, 4}}
```

Para visualizar los vectores o las matrices en la forma habitual que utilizamos en matemáticas, se utiliza el comando **MatrixForm**:

In[5]:=
```
MatrixForm[v]
```
forma de matriz

Out[5]//MatrixForm=
$$\begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$$

In[6]:=
```
MatrixForm[A]
```
forma de matriz

Out[6]//MatrixForm=
$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & -5 & 4 \end{pmatrix}$$

In[7]:=
```
MatrixForm[B]
```
forma de matriz

Out[7]//MatrixForm=
$$\begin{pmatrix} 1 & 2 \\ 3 & 0 \\ 3 & 4 \end{pmatrix}$$

Los vectores pueden sumarse (+), restarse (-), multiplicarse por un escalar (espacio en blanco o *), o multiplicarse coordenada a coordenada (espacio en blanco o *):

In[8]:=
```
u + v
```
Out[8]=
```
{3, 5, 9}
```

In[9]:=
```
3 u - 2 v
```
Out[9]=
```
{-1, 0, 2}
```

In[10]:=
```
u * v
```
Out[10]=
```
{2, 6, 20}
```

Observa la diferencia del producto anterior con el producto escalar entre vectores que se indica por el punto (.). (Debes recordar que u . v = $u_1 * v_1 + u_2 * v_2 + u_3 * v_3$)

In[11]:=
```
u.v
```
Out[11]=
```
28
```

Con respecto a las matrices, pueden sumarse (+), restarse (-), o multiplicarse término a término (espacio en blanco o *). Recuerda que para realizar estas operaciones entre dos matrices ambas han de ser de la misma dimensión:

In[12]:=
```
mB = {{1, 2}, {3, 4}};
```

In[13]:=
```
mC = {{2, 3}, {0, 3}};
```

In[14]:=
```
mB + 2 mC
```
Out[14]=
```
{{5, 8}, {3, 10}}
```

In[15]:=
```
mB * mC
```
Out[15]=
```
{{2, 6}, {0, 12}}
```

No debe confundirse la multiplicación anterior, con la multiplicación matricial, o multiplicación matriz-vector, que se indica con el punto (.) :

In[16]:=
```
mB.mC
```
Out[16]=
```
{{2, 9}, {6, 21}}
```

In[17]:=
```
A.v
```
Out[17]=
```
{3, 5, 9}
```

Recuerda que para multiplicar dos matrices sus dimensiones han de ser compatibles (el producto A.B solo es posible si el número de columnas de A coincide con el número de filas de B). Si intentamos realizar una multiplicación que no cumple estos requisitos, *Mathematica* devolverá un mensaje de error:

In[18]:=
```
mB.v
```
Dot: Tensors {{1, 2}, {3, 4}} and {2, 3, 5} have incompatible shapes.

Out[18]=
```
{{1, 2}, {3, 4}}.{2, 3, 5}
```

Para finalizar esta sección, comentaremos la forma de acceder a los elementos de una matriz o vector. Para ello, se debe escribir el nombre de la matriz o vector, seguido de la posición del elemento entre dos corchetes: **nombre de la matriz o del vector[[posicion_elemento]]**. Por ejemplo:

In[19]:=
```
A[[1, 1]]
```
Out[19]=
```
0
```

In[20]:=
```
v[[3]]
```
Out[20]=
```
5
```

In[21]:=
```
B[[2, 1]]
```
Out[21]=
```
3
```

3. Operaciones Básicas con Matrices

Mathematica incluye las funciones básicas para la manipulación de matrices: **determinante, inversa, traspuesta, rango, traza**. Veamos un ejemplo de cada una de ellas:

- Determinante:

In[22]:=
```
? Det
```
Out[22]=

Symbol

Det[m] gives the determinant of the square matrix m.

In[23]:=
```
A = {{1, 2, 3}, {3, 4, 5}, {-1, 2, 4}};
```

In[24]:=
```
Det[A]
```
determinante

Out[24]=
```
2
```

- Inversa:

In[25]:=
```
? Inverse
```
Out[25]=

Symbol

Inverse[m] gives the inverse of a square matrix m.

In[26]:=
```
invA = Inverse[A]
```
matriz inversa

Out[26]=
$$\left\{\{3, -1, -1\}, \left\{-\frac{17}{2}, \frac{7}{2}, 2\right\}, \{5, -2, -1\}\right\}$$

Recuerda que podemos utilizar el comando **MatrixForm** para mostrar la matriz:

In[27]:=
```
MatrixForm[invA]
```
forma de matriz

Out[27]//MatrixForm=

$$\begin{pmatrix} 3 & -1 & -1 \\ -\frac{17}{2} & \frac{7}{2} & 2 \\ 5 & -2 & -1 \end{pmatrix}$$

- Traspuesta:

In[28]:=
```
? Transpose
```

Out[28]=

Symbol

Transpose[*list*] transposes the first two levels in *list*.
Transpose[*list*, {n_1, n_2, …}] transposes *list* so that the k^{th} level in *list* is the n_k^{th} level in the result.
Transpose[*list*, $m \leftrightarrow n$] transposes levels m and n in *list*, leaving all other levels unchanged.

In[29]:=
```
B = {{1, 2, 4}, {2, 2, 3}};
```

In[30]:=
```
Transpose[B]
```
transposición

Out[30]=
```
{{1, 2}, {2, 2}, {4, 3}}
```

- Rango:

In[31]:=
```
? MatrixRank
```

Out[31]=

Symbol

MatrixRank[m] gives the rank of the matrix m.

In[32]:=
```
MatrixRank[B]
```
rango matricial

Out[32]=
```
2
```

- Traza:

In[33]:=
```
? Tr
```

Out[33]=

Symbol

Tr[*list*] finds the trace of the matrix or tensor *list*.
Tr[*list*, *f*] finds a generalized trace, combining terms with *f* instead of Plus.
Tr[*list*, *f*, *n*] goes down to level *n* in *list*.

In[34]:=
```
Tr[A]
```
traza

Out[34]=
```
9
```

4. Resolución de Sistemas de Ecuaciones

Para resolver sistemas de ecuaciones *Mathematica* dispone de distintas opciones. La función más general es **Solve**,

In[35]:=
```
? Solve
```

Out[35]=

Symbol

Solve[*expr*, *vars*] attempts to solve the system *expr* of equations or inequalities for the variables *vars*.
Solve[*expr*, *vars*, *dom*] solves over the domain
dom. Common choices of *dom* are Reals, Integers, and Complexes.

Esta función permite trabajar con parámetros. A continuación resolvemos la ecuación general de segundo grado:

In[36]:=
```
Solve[ a x^2 + b x + c == 0 , x]
```
resuelve

Out[36]=
$$\left\{\left\{x \to \frac{-b-\sqrt{b^2-4\,a\,c}}{2\,a}\right\},\ \left\{x \to \frac{-b+\sqrt{b^2-4\,a\,c}}{2\,a}\right\}\right\}$$

En la asignatura de Álgebra estamos interesados en sistemas de ecuaciones lineales:

Ejemplo: Resolver los siguientes sistemas de ecuaciones lineales:

$$\begin{cases} 3x+2y-z=0 \\ 2x-2y+z=1 \\ -x+2y=0 \end{cases}$$

$$\begin{cases} x+2y-z=0 \\ 2x-2y-z=1 \\ 3x-2z=1 \end{cases}$$

$$\begin{cases} x + 2y - z = 0 \\ 2x + 4y - 2z = 1 \\ -x + 2y = 0 \end{cases}$$

En el caso de sistemas, las ecuaciones deben estar agrupadas en listas (entre llaves y separadas por comas), al igual que las variables. Recordad que debemos emplear el operador de comparación ==.

In[37]:=
```
Solve[{3 x + 2 y - z == 0 , 2 x - 2 y + z == 1, -x + 2 y == 0}, {x, y, z}]
```
resuelve

Out[37]=
$$\left\{\left\{x \to \frac{1}{5}, y \to \frac{1}{10}, z \to \frac{4}{5}\right\}\right\}$$

In[38]:=
```
Solve[{x + 2 y - z == 0, 2 x - 2 y - z == 1, 3 x - 2 z == 1}, {x, y, z}]
```
resuelve

... Solve: Equations may not give solutions for all "solve" variables.

Out[38]=
$$\left\{\left\{y \to -\frac{1}{4} + \frac{x}{4}, z \to -\frac{1}{2} + \frac{3x}{2}\right\}\right\}$$

Observa que el caso anterior corresponde a un sistema compatible indeterminado. *Mathematica* avisa de tal eventualidad y expresa las variables **y, z** en términos de **x**.

In[39]:=
```
Solve[{x + 2 y - z == 0, 2 x + 4 y - 2 z == 1, -x + 2 y == 0}, {x, y, z}]
```
resuelve

Out[39]=
$$\{\}$$

En este último caso el sistema es incompatible, y *Mathematica* proporciona la lista vacía.

La función **Solve** resuelve (o al menos lo intenta) cualquier tiempo de ecuación. En el caso de sistemas lineales, es recomendable utilizar la función más específica **LinearSolve**. Para esta función los argumentos que se han de incluir en el comando son la matriz del sistema y el término independiente.

In[40]:=
```
? LinearSolve
```

Out[40]=

Symbol

LinearSolve[m, b] finds an x that solves the matrix equation $m.x == b$.
LinearSolve[m] generates a LinearSolveFunction[…] that can be applied repeatedly to different b.

Resolvemos de nuevo los sistemas anteriores:

In[41]:=
```
LinearSolve[{{3, 2 , -1} , {2, -2, 1}, {-1, 2, 0}}, {0, 1, 0}]
```
resuelve ecuación lineal

Out[41]=
$$\left\{\frac{1}{5}, \frac{1}{10}, \frac{4}{5}\right\}$$

Si el sistema es compatible indeterminado, *Mathematica* devuelve una de las soluciones (en este

caso es mejor el uso de **Solve**, o combinar esto con el comando **Nullspace**, que veremos más adelante):

In[42]:=
```
LinearSolve[{{1, 2, -1}, { 2, -2, -1}, { 3, 0, -2}}, {0, 1, 1}]
```
resuelve ecuación lineal

Out[42]=
$$\left\{\frac{1}{3}, -\frac{1}{6}, 0\right\}$$

En el caso de sistemas incompatibles, *Mathematica* anuncia que el sistema no tiene solución:

In[43]:=
```
LinearSolve[{{1, 2, -1}, {2 , +4, -2}, {-1 , 2, 0}}, {0, 1, 0}]
```
resuelve ecuación lineal

LinearSolve: Linear equation encountered that has no solution.

Out[43]=
```
LinearSolve[{{1, 2, -1}, {2, 4, -2}, {-1, 2, 0}}, {0, 1, 0}]
```

Un caso destacado de los sistemas de ecuaciones, son los sistemas homogéneos. En estos casos, se puede emplear el comando **NullSpace**, para obtener una base del espacio de soluciones (el núcleo de la aplicación lineal asociada):

Ejemplo: Resolver

$$\begin{cases} x + 2y - z = 0 \\ 3x - 2z = 0 \end{cases}$$

In[44]:=
```
NullSpace[{{1, 2, -1}, {3, 0, -2}}]
```
espacio nulo

Out[44]=
```
{{4, 1, 6}}
```

Es decir la solución del sistema anterior viene dada por: (x,y,z) = <(4,1,6)>

Para terminar esta sección comentaremos que *Mathematica* incorpora otras funciones para resolver ecuaciones (**Root**, **Reduce**, **FindRoot**, **NSolve**) de las que el lector interesado puede encontrar información detallada en la ayuda.

5. Aplicaciones Lineales

Recordamos que *Mathematica* permite definir nuestras propias funciones. La regla o prototipo para definirlas es:

In[45]:=
```
Nombre[Argumentos_] := definición
```

El guión bajo que aparece tras cada argumento, es crucial e indica a *Mathematica* que es la variable (o variables) de la ecuación que será sustituida en la definición de la derecha. Observa que antes de la definición aparece **:=**, que es el denominado operador de asignación dinámica. Este operador indica a *Mathematica* que cada vez que se encuentre el nombre de la función, debe recurrir a la definición.

Ejemplos:

In[46]:=
```
f[x_] := x^2 + 3 x - 2
```

In[47]:=
```
f[x]
```

Out[47]=
$$-2 + 3x + x^2$$

In[48]:=
```
f[3]
```

Out[48]=
```
16
```

In[49]:=
```
g[x_, y_] := (x y) / (1 + x^2 + y^2)
```

In[50]:=
```
g[1, 2]
```

Out[50]=
$$\frac{1}{3}$$

In[51]:=
```
g[a, b]
```

Out[51]=
$$\frac{a\,b}{1 + a^2 + b^2}$$

- Aplicaciones lineales:

In[52]:=
```
T[x_, y_, z_] := {x + 3 y, 2 x - y + z}
```

Observar que en este caso, dado que la función tiene valores vectoriales, utilizamos las llaves.

In[53]:=
```
T[x, y, z]
```

Out[53]=
```
{x + 3 y, 2 x - y + z}
```

In[54]:=
```
T[1, 2, 3]
```

Out[54]=
```
{7, 3}
```

Con el objetivo de ilustrar lo presentado hasta el momento proponemos el siguiente ejercicio, que resolveremos con *Mathematica*.

Ejercicio. Dada la aplicación lineal: f: $\mathbb{R}^3 \longrightarrow \mathbb{R}^2$, dada por $f(x, y, z) = (2x + 3y - z,\ 2y - 5z)$:
a) Calcular la matriz asociada en las bases canónicas.
b) Calcular bases del núcleo e imagen de f.
c) Calcular la matriz asociada en las bases {(1,0,0),(1,1,0),(1,1,1)} y {(2,3),(-2,1)}.

Definimos la aplicación lineal:

In[55]:=
```
f[x_, y_, z_] := {2 x + 3 y - z, 2 y - 5 z}
```

La matriz asociada en bases canónicas viene dada por:

In[56]:=
```
A = Transpose[{f[1, 0, 0], f[0, 1, 0], f[0, 0, 1]}]
```
transposición

Out[56]=
```
{{2, 3, -1}, {0, 2, -5}}
```

In[57]:=
```
MatrixForm[A]
```
forma de matriz

Out[57]//MatrixForm=
$$\begin{pmatrix} 2 & 3 & -1 \\ 0 & 2 & -5 \end{pmatrix}$$

Para determinar una base del núcleo usamos el comando **NullSpace**

In[58]:=
```
NullSpace[A]
```
espacio nulo

Out[58]=
```
{{-13, 10, 4}}
```

Del resultado anterior deducimos que Im f = $\mathbb{R}^2$, y por tanto cualquier base de $\mathbb{R}^2$ será base de la imagen. No obstante podemos comprobar con el comando **MatrixRank** que la dimensión del imagen es 2:

In[59]:=
```
MatrixRank[A]
```
rango matricial

Out[59]=
```
2
```

Por último para determinar la matriz asociada en las bases {(1,0,0),(1,1,0),(1,1,1)} y {(2,3),(-2,1)} utilizamos la fórmula de cambio de base:

In[60]:=
```
B = Inverse[{{2, -2}, {3, 1}}].A.{{1, 1, 1}, {0, 1, 1}, {0, 0, 1}}
```
matriz inversa

Out[60]=
$$\left\{\left\{\frac{1}{4}, \frac{9}{8}, -\frac{1}{4}\right\}, \left\{-\frac{3}{4}, -\frac{11}{8}, -\frac{9}{4}\right\}\right\}$$

In[61]:=
```
MatrixForm[B]
```
forma de matriz

Out[61]//MatrixForm=
$$\begin{pmatrix} \frac{1}{4} & \frac{9}{8} & -\frac{1}{4} \\ -\frac{3}{4} & -\frac{11}{8} & -\frac{9}{4} \end{pmatrix}$$

6. Diagonalización

En esta sección comentaremos cómo se puede determinar el polinomio característico, y calcular los autovalores y autovectores asociados a una matriz cuadrada.

Para obtener el polinomio característico *Mathematica* incorpora la función:

In[62]:=
```
? CharacteristicPolynomial
```

Out[62]=

Symbol

CharacteristicPolynomial[m, x] gives the characteristic polynomial for the matrix m.
CharacteristicPolynomial[{m, a}, x] gives the generalized characteristic polynomial with respect to a.

Por ejemplo, dada la matriz:

$$\begin{pmatrix} 7 & 0 & 0 & 0 \\ 0 & 2 & 5 & -5 \\ 13 & 0 & 7 & 8 \\ 13 & 0 & 0 & 15 \end{pmatrix}$$

In[63]:=
```
A = {{7, 0, 0, 0}, {0, 2, 5, -5}, {13, 0, 7, 8}, {13, 0, 0, 15}}
```

Out[63]=
```
{{7, 0, 0, 0}, {0, 2, 5, -5}, {13, 0, 7, 8}, {13, 0, 0, 15}}
```

su polinomio característico viene dado por:

In[64]:=
```
CharacteristicPolynomial[A, x]
```
polinomio característico

Out[64]=
$$1470 - 1253\,x + 317\,x^2 - 31\,x^3 + x^4$$

Para determinar los autovalores *Mathematica* incorpora la función:

In[65]:=
```
? Eigenvalues
```

Out[65]=

Symbol

Eigenvalues[m] gives a list of the eigenvalues of the square matrix m.
Eigenvalues[{m, a}] gives the generalized eigenvalues of m with respect to a.
Eigenvalues[m, k] gives the first k eigenvalues of m.
Eigenvalues[{m, a}, k] gives the first k generalized eigenvalues.

De modo que los autovalores de la matriz anterior son:

In[66]:=
```
Eigenvalues[A]
```
autovalores

Out[66]=
```
{15, 7, 7, 2}
```

Observa que el valor 7 aparece repetido, lo cual indica que la multiplicidad (algebraica) del valor propio 7 es 2. Aunque también se podrían calcular los autovalores determinando las raíces del polinomio característico:

In[67]:=
```
Solve[1470 - 1253 x + 317 x^2 - 31 x^3 + x^4 == 0, x]
```
resuelve

Out[67]=
```
{{x → 2}, {x → 7}, {x → 7}, {x → 15}}
```

Esta forma de proceder no es recomendable (sobre todo para dimensiones altas), pues el cálculo de raíces de un polinomio es un proceso inestable. Otra opción es tratar de factorizar el polinomio:

In[68]:=
```
Factor[1470 - 1253 x + 317 x^2 - 31 x^3 + x^4]
```
factoriza

Out[68]=
```
(-15 + x) (-7 + x)^2 (-2 + x)
```

Para obtener los autovalores podemos utilizar la función **Eigenvectors**:

In[69]:=
```
? Eigenvectors
```

Out[69]=

Symbol

Eigenvectors[m] gives a list of the eigenvectors of the square matrix m.
Eigenvectors[{m, a}] gives the generalized eigenvectors of m with respect to a.
Eigenvectors[m, k] gives the first k eigenvectors of m.
Eigenvectors[{m, a}, k] gives the first k generalized eigenvectors.

En el ejemplo que nos ocupa:

In[70]:=
```
Eigenvectors[A]
```
autovectores

Out[70]=
```
{{0, 0, 1, 1}, {-8, -13, 0, 13}, {0, 1, 1, 0}, {0, 1, 0, 0}}
```

obtenemos 4 vectores propios (luego la matriz A diagonaliza). El vector que ocupa la posición k-ésima, es el correspondiente al valor propio devuelto en la posición k-ésima por la función **Eigenvalues**. Por ejemplo, el vector (0,0,1,1) está asociado al valor propio 15, como podemos comprobar:

In[71]:=
```
A.{0, 0, 1, 1}
```

Out[71]=
```
{0, 0, 15, 15}
```

Si necesitamos obtener tanto valores como vectores propios, podemos utilizar la función **Eigensystem**:

In[72]:=
```
? Eigensystem
```

Out[72]=

Symbol

Eigensystem[m] gives a list {$values$, $vectors$}
of the eigenvalues and eigenvectors of the square matrix m.
Eigensystem[{m, a}] gives the generalized eigenvalues and eigenvectors of m with respect to a.
Eigensystem[m, k] gives the eigenvalues and eigenvectors for the first k eigenvalues of m.
Eigensystem[{m, a}, k] gives the first k generalized eigenvalues and eigenvectors.

In[73]:=
```
Eigensystem[A]
```
autovalores y autovectores

Out[73]=
```
{{15, 7, 7, 2}, {{0, 0, 1, 1}, {-8, -13, 0, 13}, {0, 1, 1, 0}, {0, 1, 0, 0}}}
```

Notar que el primer vector incorpora los valores propios y a continuación figuran los vectores propios. Para finalizar comprobemos que, efectivamente, A = P D P^{-1}, donde P es la matriz que tiene por columnas a los vectores propios y D la matriz diagonal de autovalores:

In[74]:=
```
DD = {{15, 0, 0, 0}, {0, 7, 0, 0}, {0, 0, 7, 0}, {0, 0, 0, 2}}
```

Out[74]=
```
{{15, 0, 0, 0}, {0, 7, 0, 0}, {0, 0, 7, 0}, {0, 0, 0, 2}}
```

In[75]:=
```
PP = Transpose[{{0, 0, 1, 1}, {-8, -13, 0, 13}, {0, 1, 1, 0}, {0, 1, 0, 0}}]
```
transposición

Out[75]=
```
{{0, -8, 0, 0}, {0, -13, 1, 1}, {1, 0, 1, 0}, {1, 13, 0, 0}}
```

In[76]:=
```
MatrixForm[PP. DD. Inverse[PP]]
```
forma de matriz matriz inversa

Out[76]//MatrixForm=

$$\begin{pmatrix} 7 & 0 & 0 & 0 \\ 0 & 2 & 5 & -5 \\ 13 & 0 & 7 & 8 \\ 13 & 0 & 0 & 15 \end{pmatrix}$$

7. Formas cuadráticas

En las secciones anteriores se han introducido todas las herramientas necesarias para analizar formas cuadráticas. Veamos como podríamos resolver este tipo de problemas con *Mathematica.*

Ejemplo. Clasificar la forma cuadrática:

$q(x,y,z) = 3x^2 + 3y^2 + 5z^2 - 4xy$

Introducimos su matriz asociada:

In[77]:=
```
A = {{3, -2, 0}, {-2, 3, 0}, {0, 0, 5}}
```
Out[77]=
```
{{3, -2, 0}, {-2, 3, 0}, {0, 0, 5}}
```

In[78]:=
```
Eigenvalues[A]
```
autovalores

Out[78]=
```
{5, 5, 1}
```

Dado que las tres raíces son positivas concluimos que la forma cuadrática es definida positiva.

Ejemplo. Clasificar, en función del parámetro a, la siguiente forma cuadrática

$q(x,y,z) = a\,x^2 + a\,y^2 + a\,z^2 + 2\,a\,xy$

In[79]:=
```
A = {{a, 3, 4}, {3, a, 0}, {4, 0, a}}
```
Out[79]=
```
{{a, 3, 4}, {3, a, 0}, {4, 0, a}}
```

Calculamos los autovalores

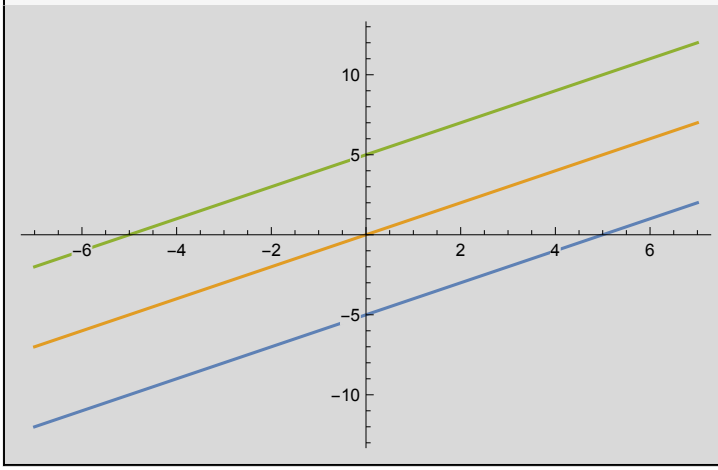
In[80]:=
```
Eigenvalues[A]
```
autovalores

Out[80]=
```
{-5 + a, a, 5 + a}
```

Dibujamos los tres valores propios,

In[81]:=
```
Plot[{-5 + a, a, 5 + a}, {a, -7, 7}]
```
representación gráfica

Out[81]=



y de aquí deducimos:

- Si a>5, q(x,y,z) es definida positiva.
- Si a= 5, q(x,y,z) es semidefinida positiva.
- Si a ϵ (-5,5), q(x,y,z) es indefinida.
- Si a = - 5, q(x,y,z) es semidefinida negativa.
- Si a< -5, q(x,y,z) es definida negativa.

8. Ejemplo. Búsqueda en la red. Algoritmo PageRank

Probablemente alguna vez te hayas preguntado cómo ordena (o indexa) un buscador las páginas que suministra tras una consulta. En realidad, el proceso es bastante complicado, pero los cinco rasgos que determinan el funcionamiento de un motor de búsqueda son:

- Análisis de los términos de búsqueda. Se trata de algoritmos de análisis de lenguaje que interpretan el significado de nuestra búsqueda.
- Búsqueda de coincidencias. Se localizan página que contengan los términos de búsqueda utilizados.
- Posicionamiento de las páginas útiles. Es la etapa fundamental, ordena las páginas según su relevancia. Este algoritmo se denomina *Page Rank* y a él dedicaremos esta sección.
- Personalización de los resultados. Se analiza el historial de búsqueda y la información personal para ofrecer mejores resultados.
- Análisis de la calidad de los resultados. Se evalúan los resultados eliminando la información redundante.

Hasta finales de los años noventa, los principales motores de búsqueda (Yahoo, Altavista, Lycos,...) basaban su funcionamiento en búsquedas de texto (coincidencias). En el año 1998, Sergey Brin y Lawrence Page, cofundadores de Google, presentan el algoritmo *PageRank*, el cual supuso una autentica revolución y fue una de las razones del extraordinario crecimiento de la multinacional estadounidense. La propuesta de Brin y Page no fue del todo original, pues otros autores habían propuesto estrategias similares en otros contextos, sin embargo, fueron ellos los que supieron darle una aplicación práctica.

El algoritmo *PageRank* ordena un conjunto de páginas web enlazadas según su relevancia. Una página se considera relevante si es enlazada por páginas relevantes. Como habrás podido comprobar, se trata de un argumento circular, que el Álgebra nos permitirá modelizar y resolver. Es más, la solución de este problema está relacionada con el cálculo de valores y vectores propios, que hemos estudiado recientemente.

Veamos cómo asigna el algoritmo *PageRank* prioridades a las páginas de una red. Asumamos que la navegación de un usuario se realiza atendiendo a la siguientes reglas:

- Con una probabilidad p, el usuario selecciona (al azar) un enlace de la página en la que se encuentra.
- Con una probabilidad (1-p), el usuario selecciona (al azar) una página de la red.

Con ello si *x* e *y,* denotan dos páginas web, la probabilidad de ir desde la página *x* a *y* es:

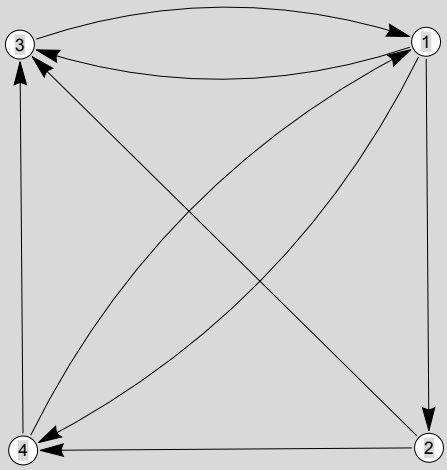
$$p(x, y) = (1-p)\frac{1}{\text{número total de webs}} + p\,\frac{\text{número de enlaces en } x \text{ que apuntan a } y}{\text{número total de enlaces en } x}$$

Según algunas fuentes, el valor de p utilizado por Google es 0.85.

Por ejemplo, dada la red:

In[82]:=
```
G = Graph[{1, 2, 3, 4}, {1 → 3, 1 → 4, 1 → 2, 2 → 4, 2 → 3, 3 → 1, 4 → 1, 4 → 3},
    grafo
  PlotTheme → "IndexLabeled"]
  tema de representación
```

Out[82]=



Si tomamos p = 0.85, obtenemos:

$$p(1,1) = 0.15 \times \frac{1}{4} \qquad p(2,1) = 0.15 \times \frac{1}{4} + 0.85\,\frac{0}{2}$$

$$p(1,2) = 0.15 \times \frac{1}{4} + 0.85\,\frac{1}{3} \qquad p(2,2) = 0.15 \times \frac{1}{4}$$

$$p(1,3) = 0.15 \times \frac{1}{4} + 0.85\,\frac{1}{3} \qquad p(2,3) = 0.15 \times \frac{1}{4} + 0.85\,\frac{1}{2}$$

$$p(1,4) = 0.15 \times \frac{1}{4} + 0.85\,\frac{1}{3} \qquad p(2,4) = 0.15 \times \frac{1}{4} + 0.85\,\frac{1}{2}$$

$$p(3,1) = 0.15 \times \frac{1}{4} + 0.85\,\frac{0}{3} \qquad p(4,1) = 0.15 \times \frac{1}{4} + 0.85\,\frac{1}{2}$$

$$p(3,2) = 0.15 \times \frac{1}{4} \qquad p(4,2) = 0.15 \times \frac{1}{4} + 0.85\,\frac{0}{2}$$

$$p(3,3) = 0.15 \times \frac{1}{4} + 0.85\,\frac{0}{3} \qquad p(4,3) = 0.15 \times \frac{1}{4} + 0.85\,\frac{1}{2}$$

$$p(3,4) = 0.15 \times \frac{1}{4} + 0.85\,\frac{0}{3} \qquad p(4,4) = 0.15 \times \frac{1}{4}$$

Escrito de forma matricial, y en forma simbólica, tenemos

In[83]:=
```
A = {{3/80, 77/240, 77/240, 77/240},
   {3/80, 3/80, 37/80, 37/80}, {71/80, 3/80, 3/80, 3/80}, {37/80, 3/80, 37/80, 3/80}};
MatrixForm[
 A]
```
forma de matriz

Out[84]//MatrixForm=

$$\begin{pmatrix} \frac{3}{80} & \frac{77}{240} & \frac{77}{240} & \frac{77}{240} \\ \frac{3}{80} & \frac{3}{80} & \frac{37}{80} & \frac{37}{80} \\ \frac{71}{80} & \frac{3}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{37}{80} & \frac{3}{80} & \frac{37}{80} & \frac{3}{80} \end{pmatrix}$$

Esta matriz recibe el nombre de *matriz de transición.* La siguiente función genera esta matriz a partir de un grafo, y del parámetro *p:*

In[85]:=
```
TransitionM[G_, p_] := Module[{n, A1, A2, A, DD, T, i, j},
                      módulo
  n = VertexCount[G];
      número de vértices
  A1 = AdjacencyMatrix[G];
       matriz de adyacencia
  T = Total[Transpose[A1]];
      total   transposición
  For[i = 1, i ≤ n, i++, If[T[[i]] == 0, A1[[i]] = Table[1, {j, 1, n}];
  para cada                 si                     tabla
    T[[i]] = n]];
  DD = DiagonalMatrix[1 / T];
       matriz diagonal
  A2 = DD.A1;
  A = (1 - p) / n + p A2;
  Return[A]]
  retorna
```

El cálculo de la matriz de transición puede hacerse a partir de la red, del siguiente modo:

In[86]:=
```
A = TransitionM[G, 85 / 100]
```

Out[86]=

$$\left\{\left\{\frac{3}{80}, \frac{77}{240}, \frac{77}{240}, \frac{77}{240}\right\}, \left\{\frac{3}{80}, \frac{3}{80}, \frac{37}{80}, \frac{37}{80}\right\}, \left\{\frac{71}{80}, \frac{3}{80}, \frac{3}{80}, \frac{3}{80}\right\}, \left\{\frac{37}{80}, \frac{3}{80}, \frac{37}{80}, \frac{3}{80}\right\}\right\}$$

Nota: Si una página no ofrece enlaces, la formula para asignar probabilidades que indicamos arriba no es válida (Observad que existe una división por cero). En ese caso, la navegación en el siguiente paso es totalmente aleatoria. Esta observación es tomada en cuenta en la línea 5 de la función anterior.

Veamos qué información nos ofrece esta matriz de transición. Supongamos que en un determinado momento un usuario se encuentra en la página web indexada con el número 3, que podemos

identificar con el vector e_0 = {0,0,1,0}. A este vector lo denominaremos *vector de estado, y* podemos interpretar cada componente como la probabilidad de que el usuario se encuentre en la página correspondiente (en este caso con probabilidad 1, está en la web 3). Observa que la multiplicación:

In[87]:=
```
{0, 0, 1, 0}.A
```

Out[87]=
$$\left\{\frac{71}{80}, \frac{3}{80}, \frac{3}{80}, \frac{3}{80}\right\}$$

o equivalentemente:

In[88]:=
```
Transpose[A].{0, 0, 1, 0}
          transposición
```

Out[88]=
$$\left\{\frac{71}{80}, \frac{3}{80}, \frac{3}{80}, \frac{3}{80}\right\}$$

proporciona otro vector de estado, e_1, cuya componente *i* – ésima proporciona la probabilidad de que el usuario se encuentre en la i-ésima página web. Observa que la multiplicación por el vector e_1 da como resultado la columna 3 de la matriz de transición, que de acuerdo con la definición indica "la probabilidad de viaje a cada nodo o web". Iterando, el vector de estado tras 10 etapas sería:

In[89]:=
```
{0, 0, 1, 0}.MatrixPower[A, 10] // N
             potencia matricial    valor numérico
```

Out[89]=
```
{0.367696, 0.142157, 0.287887, 0.20226}
```

En general, dada la matriz de transición A, y el estado inicial e_0, el estado en la etapa *n* – ésima viene dado por :

$$e_n = e_0 A^n$$

Se trata de un proceso dinámico que, como veremos, viene determinado por la matriz A. Si el tiempo de navegación tiende a infinito, el vector de estado asociado viene dado por:

$$e_\infty = \underset{n\to\infty}{\text{Limit}}\ e_0 A^n$$

Asumamos que ese límite existe, y además que es independiente del estado inicial e_0. El algoritmo *PageRank* ordena las páginas en función de los valores de ese límite. Es decir, en función de la probabilidad de que un usuario visite una página web cuando el tiempo de navegación tiende a infinito. Desde un punto de vista económico, podríamos verlo como el punto de equilibrio del sistema. Dicho esto, ¿cómo podemos calcularlo?. Observa que, utilizando las propiedades de los límites, podemos escribir:

$$e_\infty = \underset{n\to\infty}{\text{Limit}}\ e_0 A^n = \underset{n\to\infty}{\text{Limit}}\ e_0 A^{n-1} A = \left(\underset{n\to\infty}{\text{Limit}}\ e_0 A^{n-1}\right) A = e_\infty A$$

Es decir, e_∞ es vector propio asociado al valor propio 1 de la matriz A por la izquierda. O equivalente mente, el vector propio asociado al valor propio 1 de la matriz A^T por la derecha. Además, dado que e_∞ indica una probabilidad (siendo rigurosos podríamos decir que es una distribución de probabilidad discreta), sus componentes deben sumar 1.

Veamos cómo podemos calcular el *PageRank* de nuestro ejemplo utilizando las herramientas de

Mathematica:

Primero calculamos los valores propios de A^T (o A):

In[90]:=
```
Eigenvalues[Transpose[A]] // N
```
autovalores transposición valor numérico

Out[90]=
```
{1., -0.30653 + 0.349329 i, -0.30653 - 0.349329 i, -0.23694}
```

y a continuación el primer vector propio (pues el 1, aparece en la primera posición):

In[91]:=
```
e = Eigenvectors[Transpose[A], 1] // N
```
autovectores transposición va

Out[91]=
```
{{1.82182, 0.701754, 1.425, 1.}}
```

Por último, debemos normalizar este vector, es decir "convertirlo en una probabilidad", para ello dividiremos cada componete por la suma de todas ellas:

In[92]:=
```
eInfty = e[[1]] / Total[e[[1]]]
```
total

Out[92]=
```
{0.368151, 0.141809, 0.287962, 0.202078}
```

Es decir, el orden asociado a nuestro ejemplo sería: 1 > 3 > 4 > 2. De hecho, *Mathematica* incorpora una función que permite el cálculo del *PageRank* de un grafo o red:

In[93]:=
```
? PageRankCentrality
```

Out[93]=

Symbol

PageRankCentrality[g, α] gives a list of
page-rank centralities for the vertices in the graph g and weight α.
PageRankCentrality[g, α, β] gives a list of page-rank centralities,
using weight α and initial centralities β.
PageRankCentrality[{$v \to w$, ...}, ...] uses rules $v \to w$ to specify the graph g.

Observa que el parámetro α se corresponde con p en nuestra presentación. Veamos que esta función proporciona el mismo resultado que nosotros hemos obtenido:

In[94]:=
```
PageRankCentrality[G, 0.85]
```
centralidad de rango de página

Out[94]=
```
{0.368151, 0.141809, 0.287962, 0.202078}
```

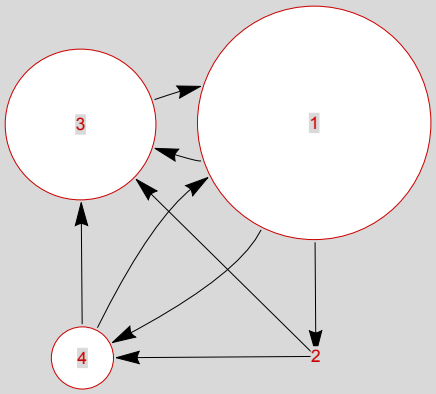
In[95]:=
```
::
```

Out[95]=
```
::2
```

Este resultado, se puede mostrar gráficamente:

In[96]:=
```
HighlightGraph[G, VertexList[G],
 destaca grafo   lista de vértices
 VertexSize → Thread[VertexList[G] → Rescale[PageRankCentrality[G, 0.85]]]]
 tamaño de vértice  atraviesa  lista de vértices   reescala  centralidad de rango de página
```

Out[96]=



Antes de incluir algún otro ejemplo revisemos alguno de los aspectos teóricos que habíamos dejado pendientes y que garantizan la existencia del *PageRank* (es decir, el límite que comentábamos antes):

- La matriz de transición, A, corresponde a un tipo de matrices denominadas no negativas. En el caso $0 \leqslant p < 1$, es además una matriz positiva.
- La matriz de transición, A, se caracteriza porque la suma de cada una de sus filas es 1. Estas matrices se denominan estocásticas por filas.
- En las condiciones anteriores, es posible demostrar que el vector propio de módulo máximo es el 1. Esto es lo que los matemáticos denominan el radio espectral de la matriz.
- Si $0 \leqslant p < 1$, la matriz A es positiva y se puede demostrar (Teorema de Perron) que la multiplicidad algebraica y geométrica del valor propio 1 de A es exactamente 1. Además, el espacio de vectores propios asociado admite un generador con todas sus componentes positivas (es por tanto un *ranking*). El generador normalizado (es decir, el múltiplo que verifica que todas sus componentes suman 1), es el *PageRank*.
- Si p=1, el problema es más complicado y omitiremos detalles. No obstante, si dadas dos páginas cualesquiera de la red siempre es posible navegar desde una hasta otra (grafo fuertemente conexo), el resultado anterior sigue siendo cierto (Teorema de Perron-Frobenius).

Teniendo en cuenta los resultados anteriores, podríamos haber calculado el *PageRank* calculando el núcleo del la aplicación asociada a la matriz $\left(A^T - \mathrm{Id}\right)$. (Recuerda cómo se realiza el cálculo del espacio propio asociado a $\lambda = 1$):

In[97]:=
```
e = NullSpace[Transpose[A] - IdentityMatrix[4]][[1]]
 espacio nulo  transposición   matriz identidad
```

Out[97]=

$$\left\{\frac{106\,613}{58\,520}, \frac{40}{57}, \frac{57}{40}, 1\right\}$$

y después normalizando el vector obtendríamos el *Page Rank*:

In[98]:=
```
eInfty = e / Total[e] // N
```
total va

Out[98]=
```
{0.368151, 0.141809, 0.287962, 0.202078}
```

Para terminar, presentamos diferentes aplicaciones construidas con *Mathematica* que permiten el cálculo del *PageRank* con las herramientas que hemos visto en el contenido de Álgebra.

In[99]:=
```
PageRankG[G_, p_] := Module[{n, A, e0, pr},
```
módulo
```
  n = VertexCount[G];
```
número de vértices
```
  A = TransitionM[G, p] // N;
```
valor numérico
```
  e0 = NullSpace[Transpose[A] - IdentityMatrix[n]][[1]];
```
espacio nulo transposición matriz identidad
```
  pr = e0 / Total[e0];
```
total
```
  Return[pr]];
```
retorna

Por ejemplo, aplicamos esta función a la red que hemos usado como hilo conductor:

In[100]:=
```
PageRankG[G, 0.85]
```

Out[100]=
```
{0.368151, 0.141809, 0.287962, 0.202078}
```

Podemos probar con una red más grande. Para ello utilizaremos la función **RandomGraph**, que nos permite generar un grafo aleatorio:

In[101]:=
```
?? RandomGraph
```

Out[101]=

Symbol

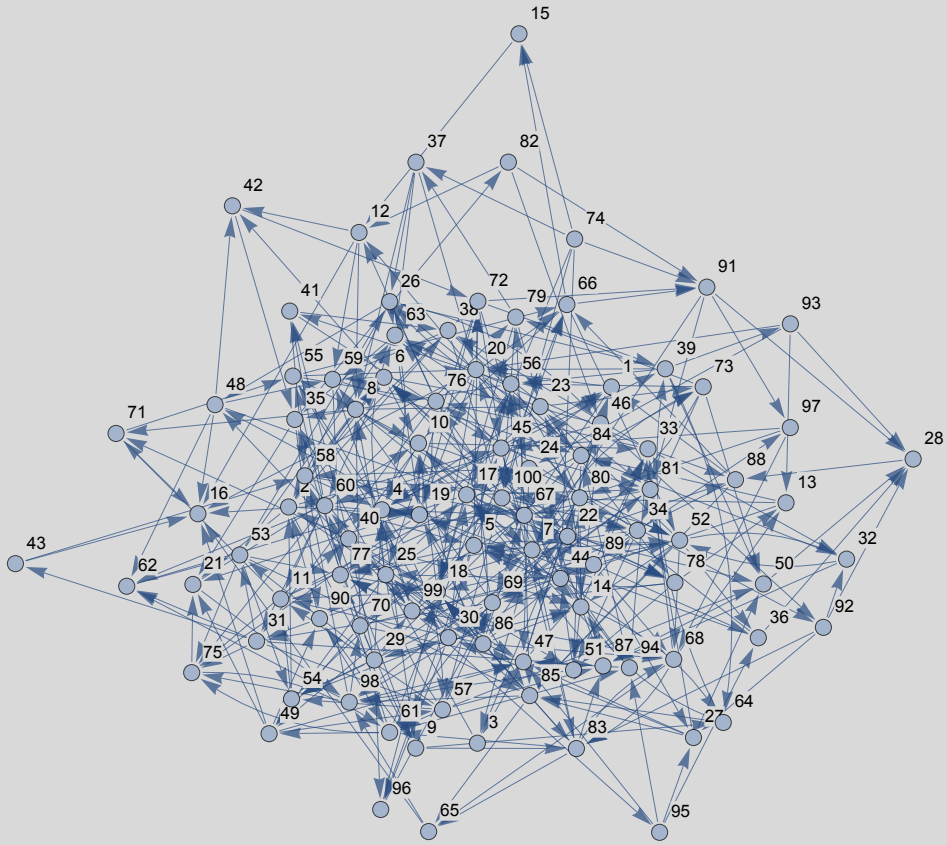
RandomGraph[{n, m}] gives a pseudorandom graph with n vertices and m edges.
RandomGraph[{n, m}, k] gives a list of k pseudorandom graphs.
RandomGraph[$gdist$, …] samples from the random graph distribution $gdist$.

Documentation Local » | Web »
Options » AlignmentPoint → Center … (59 total)
Attributes {Protected}
Full Name System`RandomGraph

In[102]:=
```
G100 = RandomGraph[{100, 450}, DirectedEdges → True];
```
grafo aleatorio aristas dirigidas verdaderc

In[103]:=
```
GraphPlot[G100]
```
representación de grafo

Out[103]=



Su *PageRank* en este caso seria:

In[104]:=
```
pr = PageRankG[G100, 0.85]
```

Out[104]=
```
{0.0154044, 0.0148626, 0.00671091, 0.0287327, 0.00807032, 0.0092806,
 0.013801, 0.00793623, 0.0104896, 0.0230716, 0.0116752, 0.0110432,
 0.00564001, 0.0140882, 0.00415929, 0.017768, 0.0112187, 0.0228637,
 0.014849, 0.0143505, 0.0155018, 0.00935129, 0.00876274, 0.00750143,
 0.00615663, 0.00976437, 0.00375069, 0.00885161, 0.00495941,
 0.00576086, 0.00947456, 0.00605273, 0.012312, 0.00935751, 0.0154339,
 0.00365363, 0.0030543, 0.00804256, 0.00642667, 0.00809622, 0.00543792,
 0.00656997, 0.00299326, 0.00826733, 0.00951238, 0.00853749,
 0.0182631, 0.00739713, 0.0107181, 0.00698627, 0.0099006, 0.0117085,
 0.00718857, 0.00878507, 0.012316, 0.0103295, 0.0122797, 0.012767,
 0.0112505, 0.00716195, 0.0131399, 0.00876925, 0.0177015, 0.00778426,
 0.00903976, 0.0107029, 0.0171505, 0.00805936, 0.0164562, 0.0076433,
 0.00690579, 0.00784821, 0.00682688, 0.00165103, 0.0139193, 0.00936383,
 0.0162679, 0.00815086, 0.00550293, 0.0150477, 0.0105511, 0.00602398,
 0.00898929, 0.0117283, 0.0155387, 0.0067841, 0.0131715, 0.0124035,
 0.00513361, 0.0132958, 0.00681908, 0.00566051, 0.00256147, 0.00434526,
 0.00216679, 0.00430749, 0.00469086, 0.0229983, 0.00546101, 0.012787}
```

Esta información en realidad no sirve de mucho, ya que es difícil determinar cual es el nodo o página “más relevante”. Sin embargo, disponemos de la función Ordering, que nos permite obtener los índices de la lista ordenada **de más a menos** relevante.

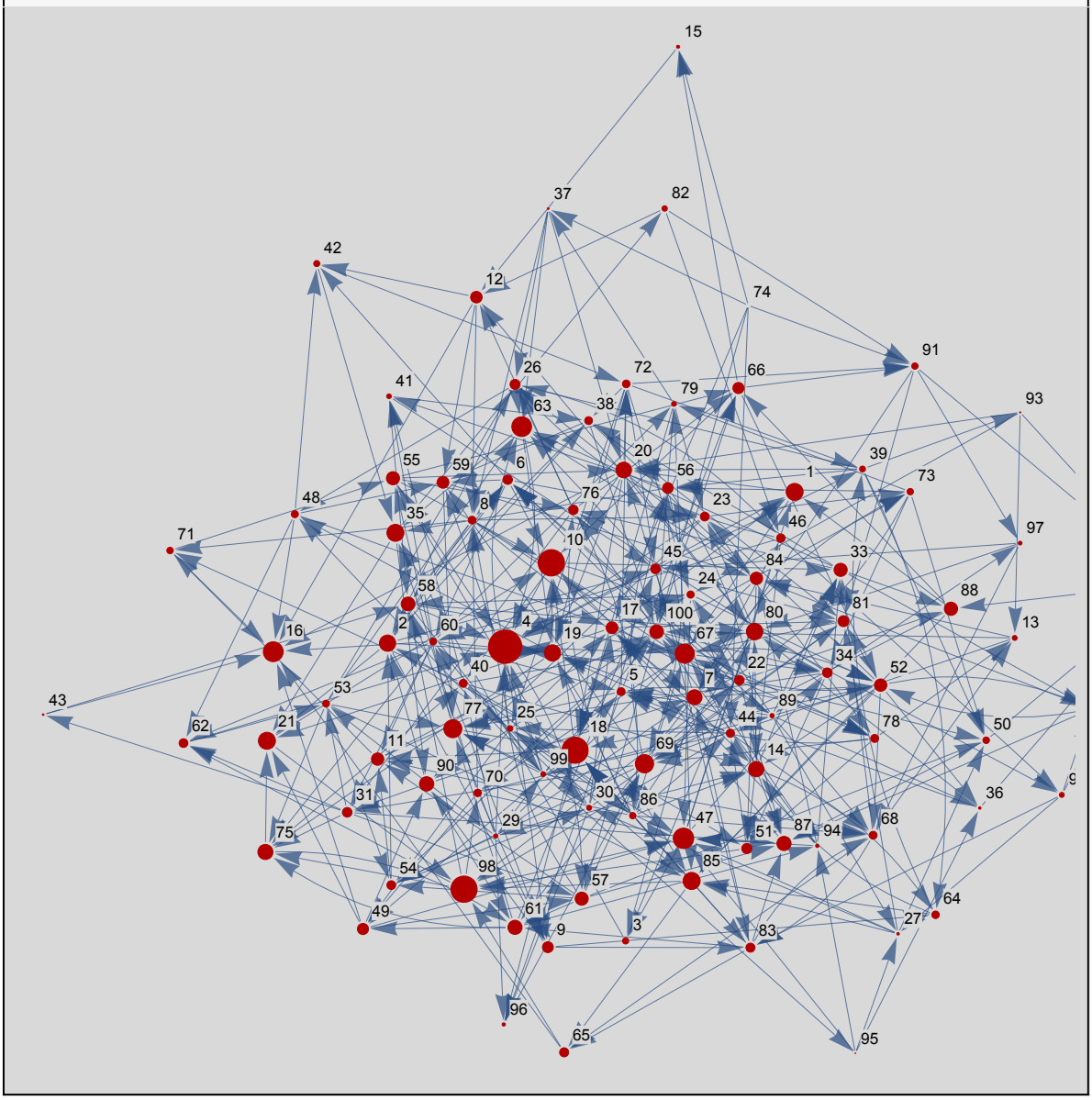
In[105]:=
```
Ordering[pr, All, Greater]
ordenación      todo  mayor
```

Out[105]=
```
{4, 10, 98, 18, 47, 16, 63, 67, 69, 77, 85, 21, 35, 1, 80, 2, 19, 20, 14, 75, 7, 90,
 87, 61, 100, 58, 88, 55, 33, 57, 84, 52, 11, 59, 17, 12, 49, 66, 81, 9, 56,
 51, 26, 45, 31, 76, 34, 22, 6, 65, 83, 28, 54, 62, 23, 46, 44, 78, 40, 5, 68,
 38, 8, 72, 64, 70, 24, 48, 53, 60, 50, 71, 73, 91, 86, 3, 42, 39, 25, 32, 82,
 30, 92, 13, 79, 99, 41, 89, 29, 97, 94, 96, 15, 27, 36, 37, 43, 93, 95, 74}
```

Esta información también la podemos visualizar:

In[106]:=
```
HighlightGraph[G100, VertexList[G100],
 VertexSize → Thread[VertexList[G100] → Rescale[pr]]]
```
destaca grafo · lista de vértices · tamaño de vértice · atraviesa · lista de vértices · reescala

Out[106]=



Con el objetivo de analizar la influencia del parámetro p, construimos la siguiente aplicación iterativa. En ella se muestra el *PageRank, el orden final, el grafo y un gráfico con el ranking* de cada página.

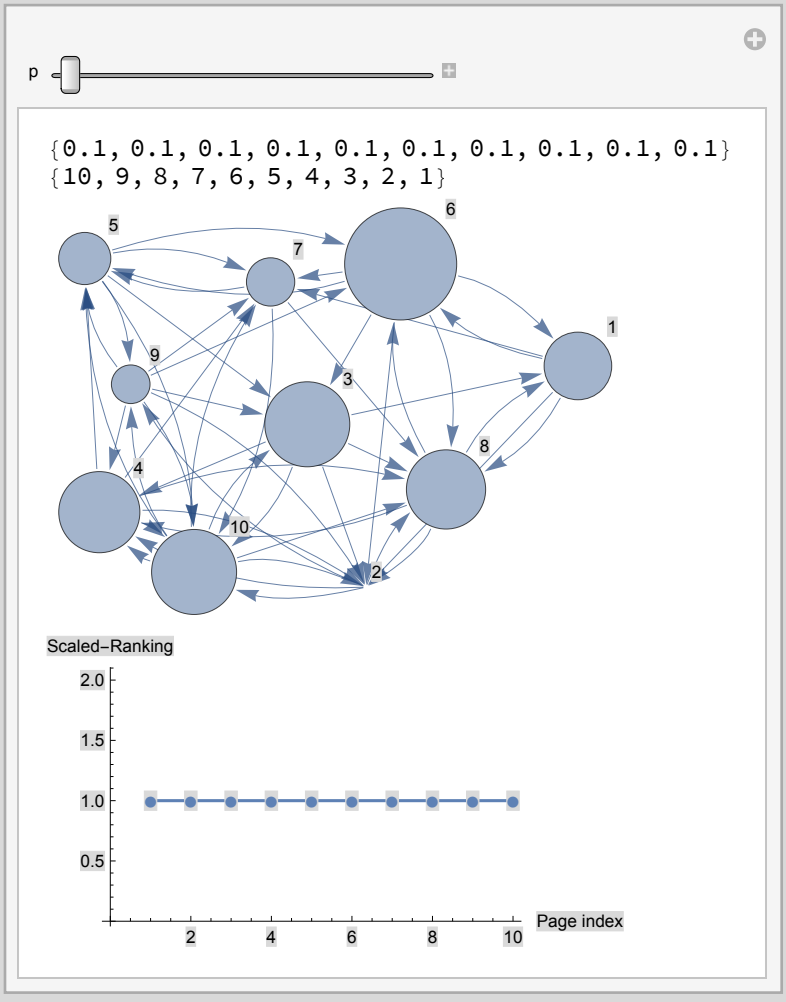
In[107]:=
```
g10 = RandomGraph[{10, 50}, DirectedEdges → True];
```
grafo aleatorio · aristas dirigidas · verdaderc

In[108]:=

```
Manipulate[Module[{pR}, Column[{pR = PageRankG[g10, p],
manipula módulo columna
                    pos = Ordering[ PageRankG[g10, p], All, Greater],
                                   ordenación todo mayor

   HighlightGraph[g10, VertexList[g10], VertexShapeFunction → "Circle",
   destaca grafo lista de vértices función de forma de vértice círculo
    VertexSize → Thread[VertexList[g10] → Rescale[pR]],
    tamaño de vértice atraviesa lista de vértices reescala
    GraphHighlightStyle → "Red", ImageSize → 300],
    estilo de destacado en grafo rojo tamaño de imagen
                    ListLinePlot[{pR / Max[pR]}, PlotMarkers → "●",
                    gráfico de línea de una li··· máximo marcadores de representación
    AxesLabel → {"Page index", "Scaled-Ranking"}, ImageSize → 300]}
    etiqueta de ejes reescalado tamaño de imagen
                ]], {p, 0, 1}]
```

Out[108]=



Nota: Observa que la función HighlightGraph no funciona correctamente con p=0, pues todos los nodos deberían ser del mismo tamaño.

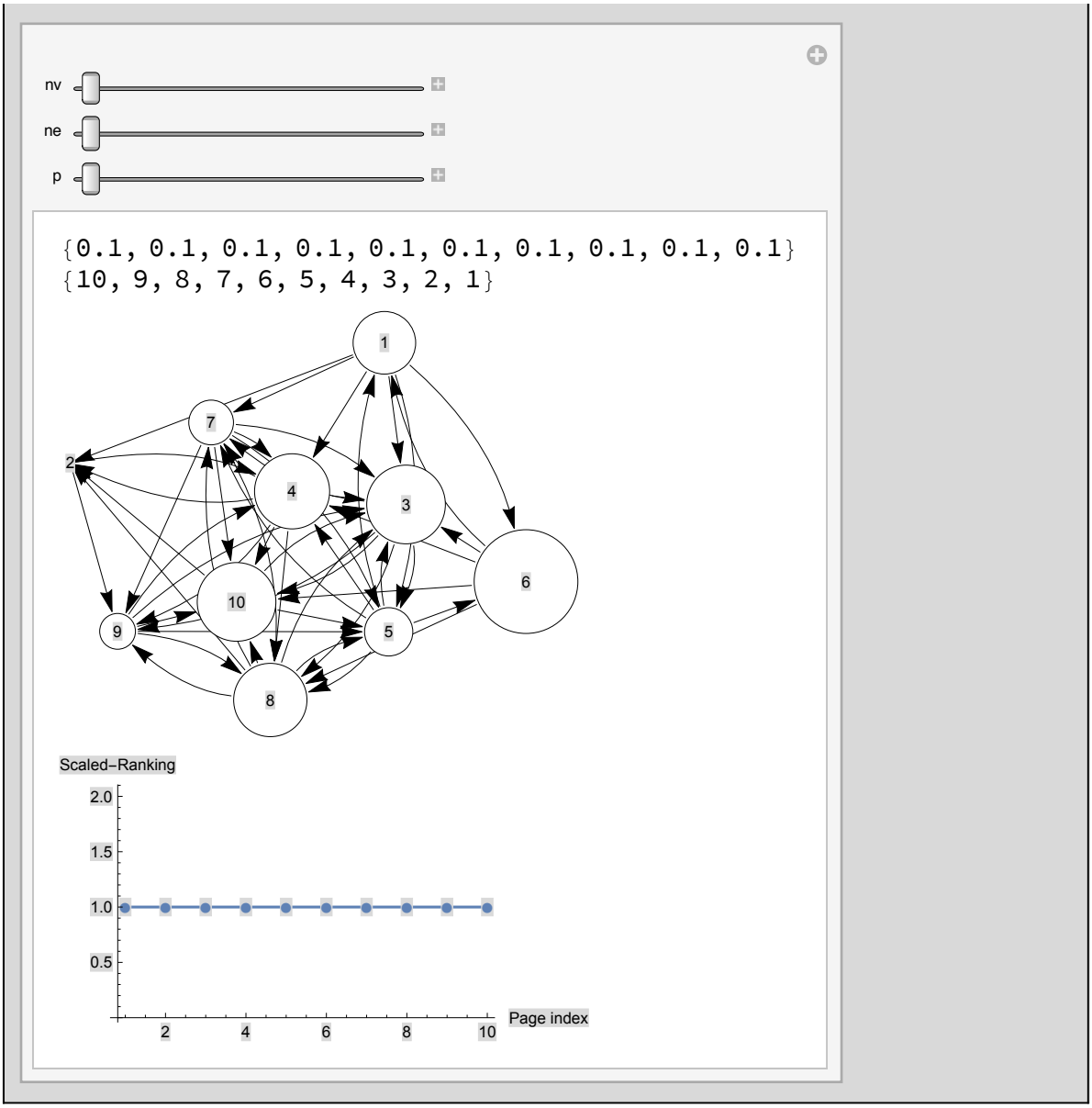
Finalmente presentamos una aplicación iterativa que genera una red aleatoria a partir del número de vértices (nv) y aristas (ne), y ordena los nodos o páginas según el criterio del *Page Rank.* La

aplicación muestra la misma información que la anterior:

In[109]:=

```
Manipulate[Module[{g, pR, ranking},
 manipula  módulo
  Column[{ne = Min[ne, nv * (nv - 1)];
   columna  mínimo
    g = RandomGraph[{Floor[nv], Floor[ne]},
        grafo aleatorio  entero inferior  entero inferior
      DirectedEdges → True, PlotTheme → "IndexLabeled"];
      aristas dirigidas  verd···  tema de representación
    Style[ pR = PageRankG[g, p], 15],
    estilo
    Style[Ordering[ PageRankG[g, p], All, Greater], 15],
    estilo  ordenación  todo  mayor
     Style[HighlightGraph[g, VertexList[g], VertexShapeFunction → "Circle",
     estilo  destaca grafo  lista de vértices  función de forma de vértice  círculo
      VertexSize → Thread[VertexList[g] → Rescale[pR]],
      tamaño de vértice  atraviesa  lista de vértices  reescala
      GraphHighlightStyle → "Red", ImageSize → 300], 50],
      estilo de destacado en grafo  rojo  tamaño de imagen
    ListLinePlot[{pR / Max[pR]}, PlotMarkers → "●", PlotRange → All,
    gráfico de línea de una li···  máximo  marcadores de represen···  rango de repr···  todo
     AxesLabel → {"Page index", "Scaled-Ranking"}, ImageSize → 300]}
     etiqueta de ejes  reescalado  tamaño de imagen
  ]
 ], {nv, 10, 50}, {ne, 50, 500}, {p, 0, 1}]
```

Out[109]=



8.1. Bibliografía

- Brin, S.; Page, L. (1998). *The anatomy of a large-scale hypertextual Web search engine*. Computer Networks and ISDN Systems. 30 (1–7): 107–117. Doi:10.1016/S0169-7552(98)00110-X.
- Sáez de Cabezón, E. *Page Rank. El algoritmo matemático que hizo a Google dominar el mundo.* https://www.youtube.com/watch?v=b3fwA3EWCd8