

Decisión de Grupo

Diego José Abengózar Vilar, z17m063
Alejandro García Castellanos, z17m008
Modelización, G-MI UPM





Índice

1. Introducción.
2. Modelización del problema.
 - a. Matrices de comparación por pares.
 - b. Escala de Saaty.
3. Modelo Analítico.
 - a. Formulación del problema de grupo.
 - b. Primeros pasos hacia la solución.
 - c. Matrices incompletas.
4. Métodos de Ajuste de Datos.
 - a. Modelo analítico de ajuste de datos.
 - b. Modelo computacional:
 - i. Métrica 2
 - ii. Métrica 1
 - c. Medidas de análisis de la solución.
 - d. Ejemplos
5. Conclusiones.
6. Referencias.
7. Anexo

Introducción



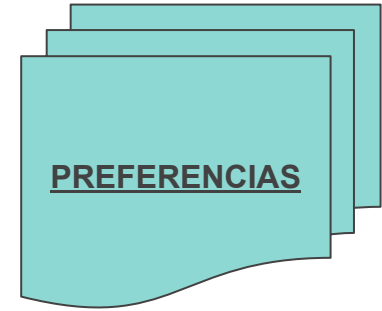
Problema de decisión de grupo

- Un problema que aparece en multitud de campos (teoría de la decisión, sistemas recomendadores...) es el denominado problema de decisión de grupo.
- El objetivo es ordenar un conjunto de alternativas A_1, \dots, A_n atendiendo a las preferencias de uno o varios expertos individuales (E_1, \dots, E_m).



¿Qué es el problema de decisión de grupo?

- Atendiendo a la complejidad de evaluar simultáneamente un gran número de alternativas, consideramos que los expertos expresan sus preferencias mediante matrices de comparación por pares (M^i).
- Para resolver el problema, calcularemos un vector de pesos, w , que indique cómo de prioritaria es cada una de las opciones.



MÉTODOS



$$w = (w_1, \dots, w_n)^t$$

Vector de prioridad:

- w_i indica la prioridad de la alternativa A_i
- A mayor w_i mayor prioridad



Modelización del problema





Matrices de comparación por pares (MCP)

- Cada experto representa sus preferencias entre una alternativa y otra a través de una matriz de comparación por pares

$$M = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \dots & \dots & \dots & \dots \\ m_{n1} & m_{n2} & \dots & m_{nn} \end{pmatrix}$$

- El elemento m_{ij} de la matriz M es una estimación de la razón de importancia entre la alternativa A_i y A_j .



Escala de Saaty

- Para cuantificar la información de los expertos utilizaremos la Escala de Saaty: los expertos expresan la importancia de una opción frente a otra mediante un número del 1 al 9 (y sus recíprocos).
- Se basa en estudios psicológicos que muestran que un individuo no puede comparar simultáneamente más de 7 (± 2) objetos.



Thomas L. Saaty



Escala de Saaty

Escala	Definición	Explicación
1	Igual importancia	Los dos elementos contribuyen igualmente al objetivo.
3	Importancia moderada	La experiencia y el juicio están ligeramente a favor de uno de los elementos.
5	Importancia fuerte	La experiencia y el juicio están fuertemente a favor de uno de los elementos.
7	Importancia muy fuerte o demostrable	Un elemento es preferido sobre el otro en un grado muy fuerte y esta preferencia puede demostrarse en la práctica.
9	Importancia absoluta	La evidencia favorece a una alternativa sobre la otra extremadamente.
2, 4, 6, 8	Valores intermedios	Algunas veces se necesita interpolar un juicio, porque no hay una palabra que describa la relación entre los elementos.



Propiedades de la MCP

- **Propiedades:**
 1. Matrices cuadradas de orden n , siendo n el número de alternativas a comparar
 2. Todas sus componentes son positivas, $m_{ij} > 0 \forall i, j$.
 3. **Reciprocidad:** $m_{ij} * m_{ji} = 1 \forall i, j$.
 4. Los elementos de la diagonal son 1, $m_{ii} = 1 \forall i$.
- Si la matriz cumple la **propiedad de consistencia**, $m_{ij} * m_{jk} = m_{ik} \forall i, j, k$, decimos que la matriz es consistente.



Índice de consistencia

La matriz de comparación por pares puede no ser consistente y para medir cómo de consistente es la matriz se utiliza un **índice de consistencia**:

$$IC(M) = \frac{\lambda_{max} - n}{n - 1}$$

(n es la dimensión de M)
(λ_{max} autovalor dominante de M)

- El aumento del valor del índice de consistencia indica un mayor grado de inconsistencia de los juicios recogidos en la matriz.
- **Si M es consistente, su índice de consistencia es 0.**



Modelo analítico





Formulación del problema de grupo

Datos de entrada:

- M^1, \dots, M^m MCP de dimensión $n \times n$.
- Recíprocas.
- Datos en escala Saaty.
- Pueden ser **incompletas**.
- Pueden ser **inconsistentes**.
- Pueden ser **discrepantes**.

Objetivo:

- Vector de pesos del grupo w .
- Inducir un ranking en las opciones A_i a partir de w .

$$\begin{cases} E_1 \Rightarrow M^1 = (m_{ij}^1)_{i,j=1..n} \\ \dots \\ E_m \Rightarrow M^m = (m_{ij}^m)_{i,j=1..n} \end{cases}$$

$$\rightarrow \text{METODOS} \rightarrow w = (w_1, \dots, w_n)^t$$



Primeros pasos hacia la solución

Teorema (Saaty): Si M es una matriz consistente, entonces:

- Autovalores de M : 0 y n .
- Existe un vector positivo $w = (w_1, \dots, w_n)^t$ tal que $m_{ij} = w_i / w_j$, vector de pesos que buscamos.

Además w^t es el único autovector asociado al autovalor dominante de $M(n)$ tal que $\sum_{i=1}^n w_i = 1$

- Si M es recíproca,
 - Autovalor dominante de M $\lambda_{max} \geq n$
 - M consistente $\Leftrightarrow \lambda_{max} = n$



Primeros pasos hacia la solución


De forma que si la matriz es consistente se cumple que existe un vector w positivo tal que:

$$M^k = \begin{pmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\ \frac{w_3}{w_1} & \frac{w_3}{w_2} & \cdots & \frac{w_3}{w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & \frac{w_n}{w_n} \end{pmatrix}$$

Así que, si la matriz es consistente podemos obtener la solución a través del sistema no lineal de ecuaciones, para cada k :

$$\frac{w_i}{w_j} = m_{ij}^k \quad \forall i, j, k$$

Y si queremos que la solución sea única podemos normalizar el vector de forma que $\sum w_i = 1$



En una MCP los elementos son mayores que
cero.



Usamos el valor 0 como indicador de que no
se tiene esa información de dicho experto



No añadimos su ecuación $\frac{w_i}{w_j} = m_{ij}^k$ al
sistema

Matrices incompletas

Hay veces que los decisores no
proporcionan un dato porque no
tienen información precisa sobre
alguna de las opciones.



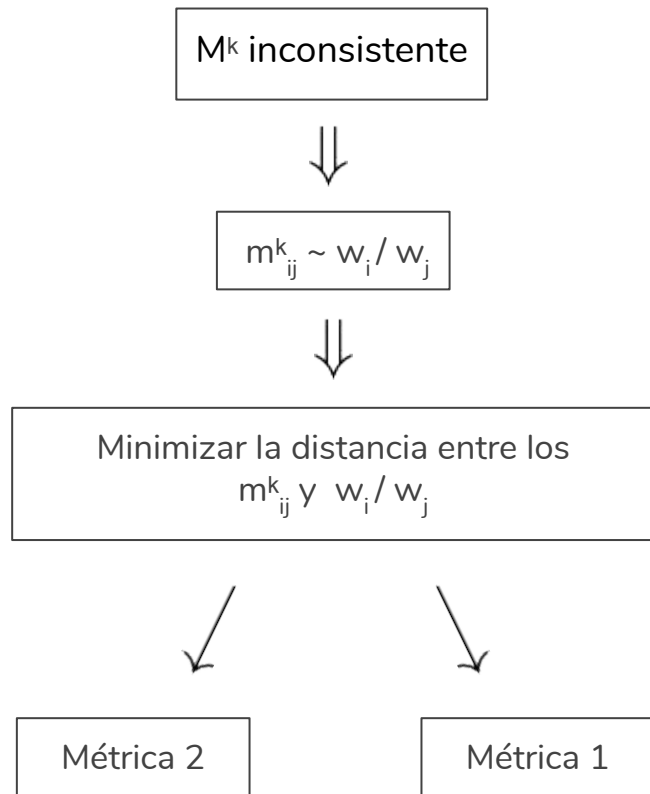


Métodos de ajuste de datos



¿Cómo obtenemos el vector de prioridad cuando no es consistente?

En los problemas reales, atendiendo a la complejidad del problema y la subjetividad inherente a los juicios humanos, las matrices de comparación por pares pueden no ser consistentes.





Modelo analítico de ajuste de datos

Se buscan w_1, \dots, w_n positivos que mejor ajustan los datos m_{ij}^k teniendo en cuenta todos los expertos, para la métrica p :

$$\text{Min} \sum_{k=1}^m \sum_{i,j=1}^n \left| m_{ij}^k - \frac{w_i}{w_j} \right|^p \quad \text{si } 1 \leq p < \infty$$

Estudiaremos el problema para las métricas más usuales, $p = 1, 2$



Modelo Computacional

Métrica 2

Método de mínimos cuadrados



Métrica 2

Se buscan los w_1, \dots, w_n positivos que mejor ajustan los datos m_{ij}^k teniendo en cuenta todos los expertos, en el sentido de mínimos cuadrados:

$$\text{Min} \sum_{k=1}^m \sum_{i=1}^n \sum_{j=1}^n \left(m_{ij}^k - \frac{w_i}{w_j} \right)^2$$

Para resolver la **no linealidad**, utilizaremos dos técnicas para linealizar el problema:

- Transformación logarítmica
- Problema ponderado



Métrica 2 - Transformación logarítmica

1. Partiendo de las ecuaciones **no lineales**: $m_{ij}^k - \frac{w_i}{w_j} = 0 \quad \forall i, j = 1..n, k = 1..m$

2. **Tomamos logaritmos**: $\log(w_i) - \log(w_j) = \log(m_{ij}^k) \quad \forall i, j = 1..n, k = 1..m$

3. Resultando las ecuaciones **lineales**: $l_{ij}^k - v_i + v_j \quad \forall i, j = 1..n, k = 1..m$

4. Se obtiene el sistema lineal sobredeterminado para cada experto:

$$B^k = \begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -1 & 0 & 0 & \dots & 1 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & -1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} -l_{12}^k \\ -l_{13}^k \\ \dots \\ -l_{1n}^k \\ -l_{23}^k \\ \dots \\ -l_{2n}^k \\ \dots \\ -l_{n-1,n}^k \end{pmatrix} = b^k$$



Métrica 2 - Transformación logarítmica

Resolvemos el sistema lineal sobredeterminado mediante **mínimos cuadrados**:

$$\begin{pmatrix} B^1 \\ B^2 \\ \dots \\ B^m \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} b^1 \\ b^2 \\ \dots \\ b^m \end{pmatrix}$$

1. Deshacemos la transformación logarítmica: $w_i = e^{v_i} \quad \forall i = 1, \dots, n$
2. Normalizamos: $w_i = \frac{w_i}{\sum_1^n w_i} \quad \forall i = 1, \dots, n$
3. En w obtenemos un vector con los pesos de prioridad de las alternativas.



Métrica 2 - Problema Ponderado

1. Partiendo de las ecuaciones **no lineales**: $m_{ij}^k - \frac{w_i}{w_j} = 0 \quad \forall i, j = 1..n, k = 1..m$
2. Se multiplica por w_j para obtener la ecuación **lineal**: $m_{ij}^k \cdot w_j - w_i = 0 \quad \forall i, j, k$
3. Obtenemos el siguiente sistema para cada experto:

$$B^k = \begin{pmatrix} 1 & -m_{12}^k & 0 & \dots & 0 \\ 1 & 0 & -m_{13}^k & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & \dots & 0 & -m_{1n}^k \\ -m_{21}^k & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & 0 & \dots & -m_{2n}^k \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = b^k$$



Métrica 2 - Problema Ponderado

1. Calculamos la solución de mínimos cuadrados del sistema lineal sobredeterminado:

$$\begin{pmatrix} B^1 \\ B^2 \\ \dots \\ B^m \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} b^1 \\ b^2 \\ \dots \\ b^m \end{pmatrix}$$

2. Normalizamos: $w_i = \frac{w_i}{\sum_1^n w_i} \quad \forall i = 1, \dots, n$
3. En w obtenemos un vector con los pesos de prioridad de las alternativas

Métrica 1

Método que minimiza la métrica vectorial 1



Métrica 1

Se buscan los w_1, \dots, w_n positivos que mejor ajustan los datos m_{ij}^k en el sentido que minimicen la métrica vectorial 1:

$$\text{Min} \sum_{k=1}^m \sum_{i=1}^n \sum_{j=1}^n \left| m_{ij}^k - \frac{w_i}{w_j} \right|$$

Para resolver la **no linealidad**, utilizaremos dos técnicas para linealizar el problema:

- Transformación logarítmica
- Problema ponderado

No es diferenciable, así que lo transformaremos a un problema equivalente de programación lineal introduciendo nuevas variables.

Métrica 1 - Transformación logarítmica

1. Aplicando la transformación logarítmica obtenemos:
$$\text{Min} \sum_{k=1}^m \sum_{i=1}^n \sum_{j=1}^n |l_{ij}^k - v_i + v_j|$$

2. Introducimos las variables n_{ij}^k y p_{ij}^k :

$$n_{ij}^k = \frac{1}{2} [|v_i - v_j - l_{ij}^k| + (v_i - v_j - l_{ij}^k)] \quad \text{Desviación negativa}$$

$$p_{ij}^k = \frac{1}{2} [|v_i - v_j - l_{ij}^k| - (v_i - v_j - l_{ij}^k)] \quad \text{Desviación positiva}$$

Se observa que:
$$n_{ij}^k + p_{ij}^k = |l_{ij}^k - v_i + v_j|$$

$$n_{ij}^k - p_{ij}^k = v_i - v_j - l_{ij}^k$$

$$n_{ij}^k \geq 0, p_{ij}^k \geq 0, n_{ij}^k \cdot p_{ij}^k = 0 \quad (\text{Si uno es distinto de cero el otro es cero})$$



Métrica 1 - Transformación logarítmica

El problema de programación lineal por metas resultante es:

$$\min D = \sum_{k=1}^m \sum_{i=1}^n \sum_{j=1}^n (n_{ij}^k + p_{ij}^k)$$

s.a

$$l_{ij}^k - v_i + v_j + n_{ij}^k - p_{ij}^k = 0$$

$$n_{ij}^k \geq 0, p_{ij}^k \geq 0 \quad \forall i, j, k$$

1. Tomamos v del vector solución obtenido a través del método del simplex.
2. Deshacemos la transformación logarítmica: $w_i = e^{v_i} \quad \forall i = 1, \dots, n$
3. Normalizamos: $w_i = \frac{w_i}{\sum_1^n w_i} \quad \forall i = 1, \dots, n$



Métrica 1 - Problema Ponderado

1. Multiplicando por w_j :

$$\text{Min} \sum_{k=1}^m \left(\sum_{i=1}^n \sum_{j=1}^n |m_{ij}^k \cdot w_j - w_i| + \sum_{i=1}^n |w_i - 1| \right)$$

2. Introducimos las variables n_{ij}^k y p_{ij}^k :

$$n_{ij}^k = \frac{1}{2} [|m_{ij}^k \cdot w_j - w_i| + (m_{ij}^k \cdot w_j - w_i)] \quad \text{Desviación negativa}$$

$$p_{ij}^k = \frac{1}{2} [|m_{ij}^k \cdot w_j - w_i| - (m_{ij}^k \cdot w_j - w_i)] \quad \text{Desviación positiva}$$

Se observa que: $|m_{ij}^k \cdot w_j - w_i| = n_{ij}^k + p_{ij}^k$

$$n_{ij}^k - p_{ij}^k = m_{ij}^k \cdot w_j - w_i$$

$$n_{ij}^k \geq 0, \quad p_{ij}^k \geq 0, \quad n_{ij}^k \cdot p_{ij}^k = 0$$



Métrica 1 - Problema Ponderado

El problema de programación lineal por metas resultante es:

$$\min D = \sum_{k=1}^m \sum_{i=1}^n \sum_{j=1}^n (n_{ij}^k + p_{ij}^k)$$

$$\text{s.a} \quad w_i - m_{ij}^k \cdot w_j + n_{ij}^k - p_{ij}^k = 0$$
$$n_{ij}^k \geq 0, p_{ij}^k \geq 0 \quad \forall i, j, k$$

1. Tomamos w del vector solución obtenido a través del método del simplex.
2. Normalizamos: $w_i = \frac{w_i}{\sum_1^n w_i} \quad \forall i = 1, \dots, n$



Medidas para analizar la solución





Errores de grupo

$$AcuerdoGrupo_{Fr} = \left\{ \sum_{k=1:m_{ij}^k: dato}^m \sum_{i=1}^n \sum_{j=1}^n \left(m_{ij}^k - \frac{w_i}{w_j} \right)^2 \right\}^{1/2}$$

Métrica 2

$$AcuerdoGrupo_1 = \left\{ \sum_{k=1:m_{ij}^k: dato}^m \sum_{i=1}^n \sum_{j=1}^n \left| m_{ij}^k - \frac{w_i}{w_j} \right| \right\}$$

Métrica 1

$$MaximoDesacuerdoGrupo = \max \left\{ \left| m_{ij}^k - \frac{w_i}{w_j} \right| \right\}$$

Métrica Inf.

- Sólo se calculan los residuos y la agregación de éstos para los datos conocidos.
- Los errores se relativizan al n° de datos conocidos.



Errores de individuales

Errores producidos por la solución w respecto de la información de preferencias dada por cada experto:

$$\begin{array}{ccc} E^1 & \dots & E^k & \dots & E^m \\ \downarrow & & \downarrow & & \downarrow \\ \|M^1 - W\| & & \|M^k - W\| & & \|M^m - W\| \end{array}$$



Ejemplos





Consistente, mismas opiniones

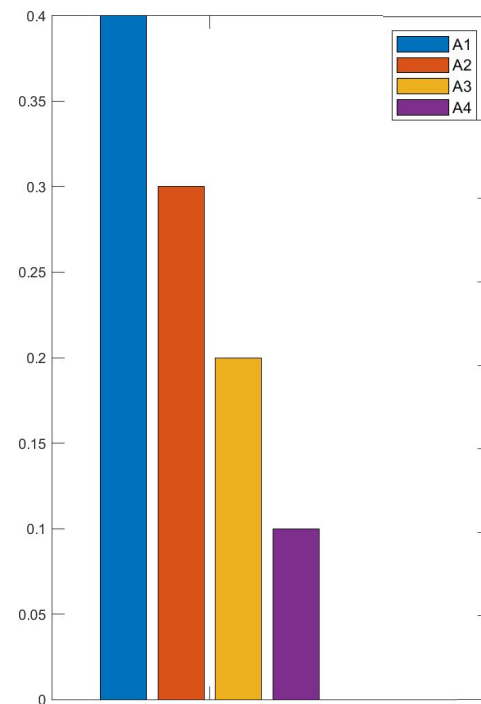
$$E_1 \longrightarrow M^1 = \begin{pmatrix} 1.0000 & 1.3333 & 2.0000 & 4.0000 \\ 0.7500 & 1.0000 & 1.5000 & 3.0000 \\ 0.5000 & 0.6667 & 1.0000 & 2.0000 \\ 0.2500 & 0.3333 & 0.5000 & 1.0000 \end{pmatrix}$$

$$E_2 \longrightarrow M^2 = \begin{pmatrix} 1.0000 & 1.3333 & 2.0000 & 4.0000 \\ 0.7500 & 1.0000 & 1.5000 & 3.0000 \\ 0.5000 & 0.6667 & 1.0000 & 2.0000 \\ 0.2500 & 0.3333 & 0.5000 & 1.0000 \end{pmatrix}$$

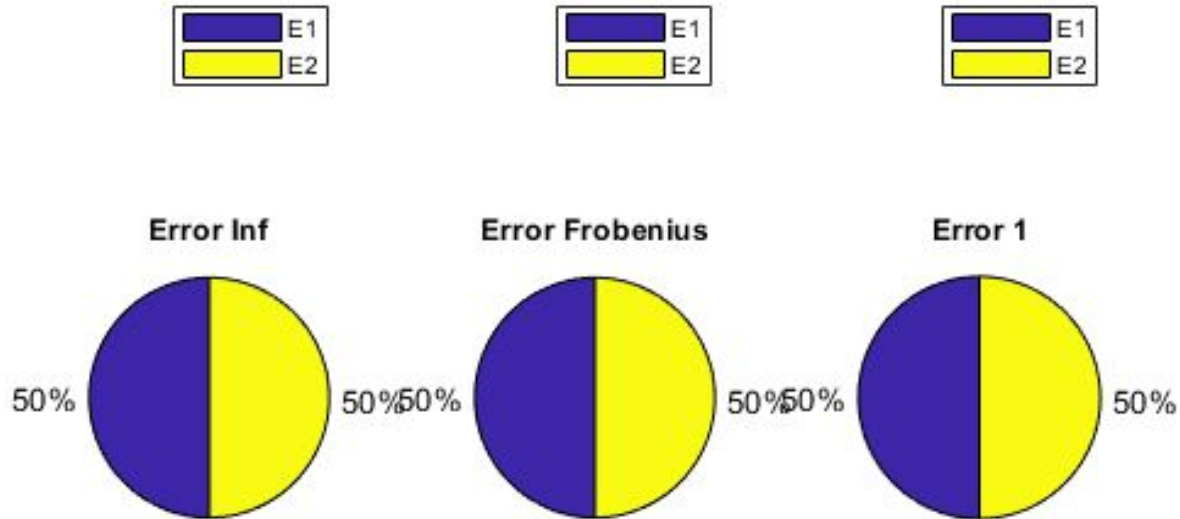


Resultados

	w	Ranking
Min Cuad Log	(0.4000,0.3000,0.2000,0.1000)	A1 > A2 > A3 > A4
Min Cuad Pond	(0.4000,0.3000,0.2000,0.1000)	A1 > A2 > A3 > A4
Min Sum Des Log	(0.4000,0.3000,0.2000,0.1000)	A1 > A2 > A3 > A4
Min Sum Des Pond	(0.4000,0.3000,0.2000,0.1000)	A1 > A2 > A3 > A4



Comparación de los errores individuales

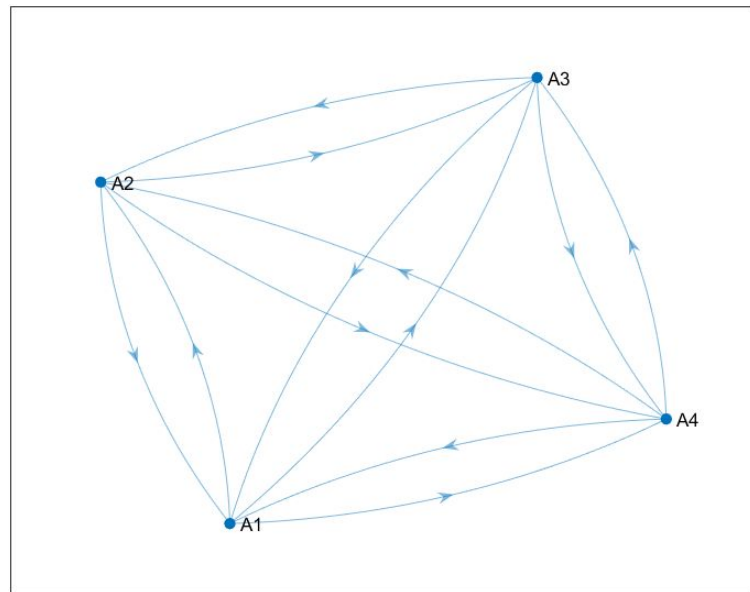


Inconsistente: varias opiniones e incompleta

$$E_1 \rightarrow M^1 = \begin{pmatrix} 1.0000 & 0 & 0.1429 & 0.2000 \\ 0 & 1.0000 & 0.5000 & 0.3333 \\ 7.0000 & 2.0000 & 1.0000 & 0.1111 \\ 5.0000 & 3.0000 & 9.0000 & 1.0000 \end{pmatrix}$$

$$E_2 \rightarrow M^2 = \begin{pmatrix} 1.0000 & 0 & 0.3333 & 0.1111 \\ 0 & 1.0000 & 0 & 0.1250 \\ 3.0000 & 0 & 1.0000 & 0.1111 \\ 9.0000 & 8.0000 & 9.0000 & 1.0000 \end{pmatrix}$$

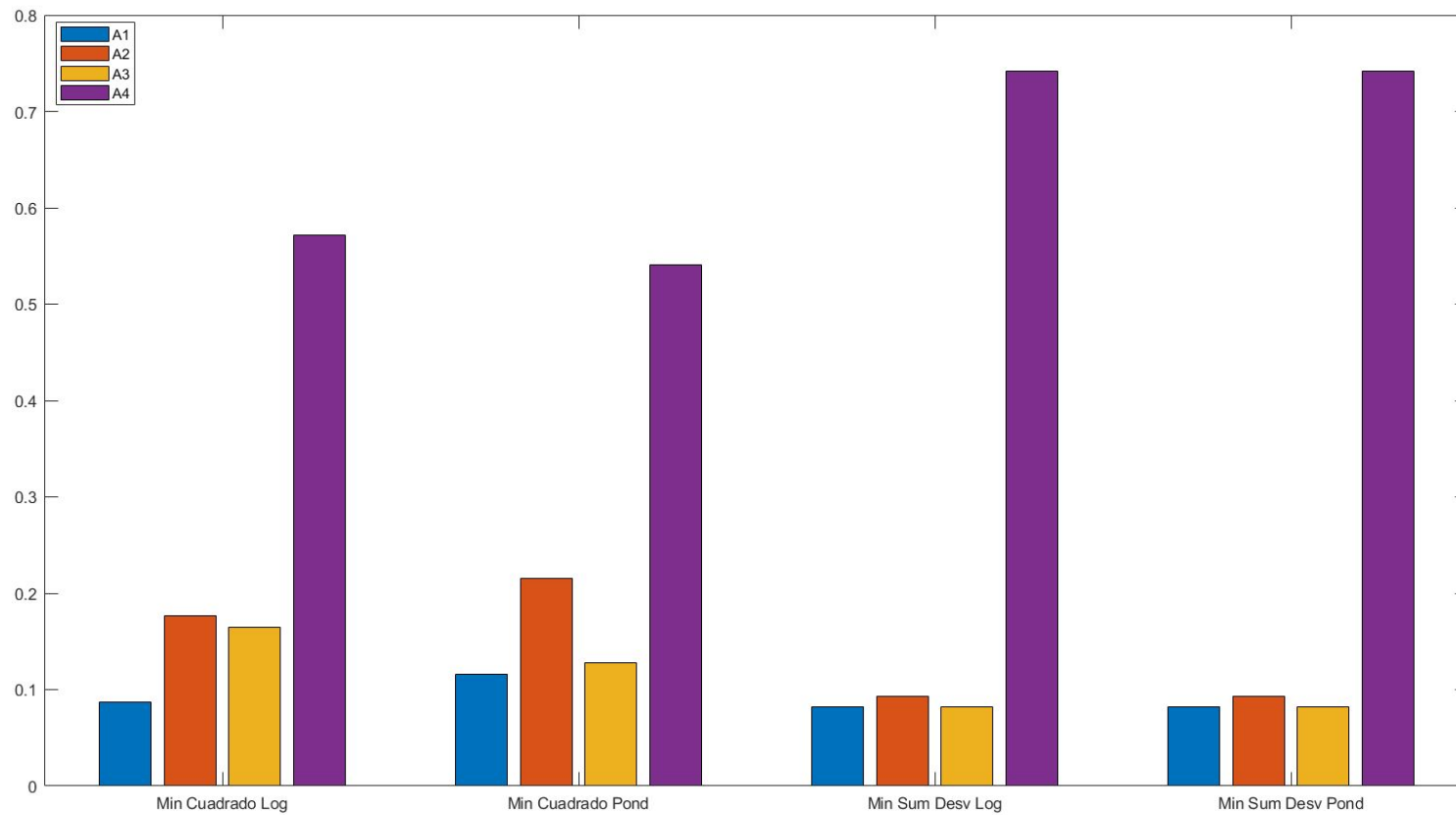
$$E_3 \rightarrow M^3 = \begin{pmatrix} 1.0000 & 3.0000 & 0 & 0 \\ 0.3333 & 1.0000 & 2.0000 & 0.2000 \\ 0 & 0.5000 & 1.0000 & 0 \\ 0 & 5.0000 & 0 & 1.0000 \end{pmatrix}$$





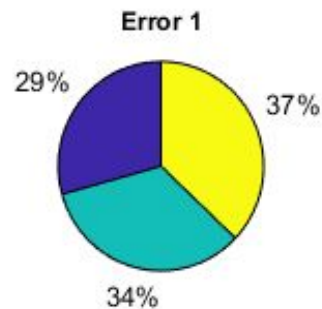
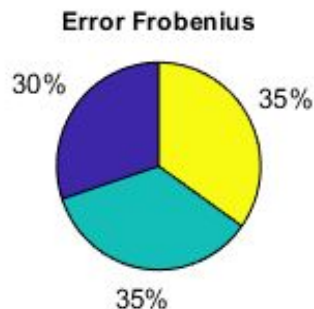
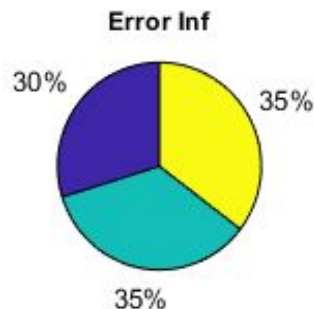
Resultados

	w	Ranking	Max Residuo	Error Fr	Error Norm1
Min Cuad Log	(0.0870,0.1769,0.1648,0.5714)	A4 > A2 > A3 > A1	0.5533	1.2761	4.2437
Min Cuad Pond	(0.1161,0.2154,0.1276,0.5409)	A4 > A2 > A3 > A1	0.5901	1.3135	4.3932
Min Sum Des Log	(0.0825,0.0928,0.0825,0.7423)	A4 > A2 > A3 = A1	0.7800	1.3208	3.7399
Min Sum Des Pond	(0.0825,0.0928,0.0825,0.7423)	A4 > A2 > A3 = A1	0.7800	1.3208	3.7399

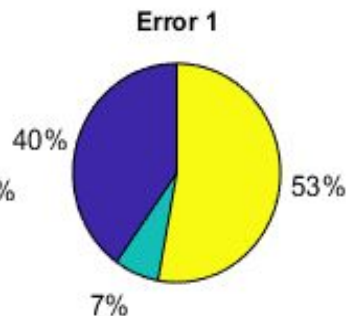
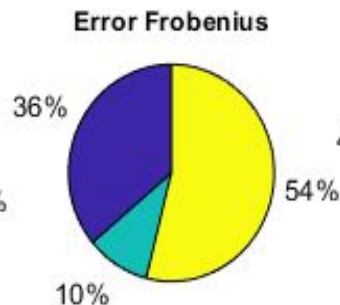
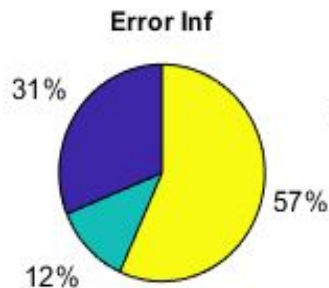


Comparación de los errores individuales

Min Cuad



Min Sum
Desv





Conclusiones



Problema



MÉTODOS

$$w = (w_1, \dots, w_n)^t$$

Obstáculos

**Matrices
inconsistentes**

**Matrices
incompletas**

**Diferentes
preferencias**

MODELO ANALÍTICO



MÉTODOS
COMPUTACIONALES



IMPLEMENTACIÓN



Referencias





Referencias

- Dopazo, E & González Pachón, J. (2003). Consistency-driven approximation of a pairwise comparison matrix. *Kybernetika* 39(5), 561-568.
- Dopazo, E & Ruiza-Tangle, M. (2011). A parametric GP model dealing with incomplete information for group decision-making. *Applied Mathematics and Computation*, 218(2), 514-519.
- Saaty, T. L. (2003). Decision-making with the AHP: Why is the principal eigenvector necessary. *European Journal of Operational Research*, 145(1), 85-91.
- Ueberhuber, C.W. *Numerical Computation 1 Methods, Software, and Analysis*. Springer.
- Peláez, J.I. & Lamata, M.T. (2003). A new measure of consistency for positive reciprocal matrices, *Computers & Mathematics with Applications*, Volume 46, Issue 12, 1839-1845.

Anexo



ANEXO:

casosPrueba.m

```
clc;
clear;

fprintf("-----Consistente-----\n");
w = [0.4 0.3 0.2 0.1]
M= zeros(4);
for i = 1:4
    for j = 1:4
        M(i,j) = w(i)/w(j);
    end
end

M
ic(M)

fprintf("-----Metodo de la potencia-----\n");
w0 = funciones(0, M) % potencia(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w0, M)

fprintf("-----Metodo de Minimos Cuadrados [Log]-----\n");
w1 = funciones(1, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w1, M)

fprintf("-----Metodo de Minimos Cuadrados [Ponderado]-----\n");
w2 = funciones(2, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w2, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Log]-----\n");
w3 = funciones(3, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w3, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Ponderado]-----\n");
w4 = funciones(4, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w4, M)

%Dibujar pesos
figure();
c = categorical(["Potencia", "Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv Pond"]);
bar(c, [w0';w1';w2';w3';w4']);
l=compose("A%d", (1:length(M)));
legend(l);

clear;
```

```

fprintf("-----Consistente Varios Expertos-----\n");
w = [0.4 0.3 0.2 0.1]
M= zeros(4);
for i = 1:4
    for j = 1:4
        M(i,j) = w(i)/w(j);
    end
end

M
ic(M)

fprintf("-----Metodo de Minimos Cuadrados [Log]-----\n");
w1 = funciones(1, M, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w1, M, M)

fprintf("-----Metodo de Minimos Cuadrados [Ponderado]-----\n");
w2 = funciones(2, M, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w2, M, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Log]-----\n");
w3 = funciones(3, M, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w3, M, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Ponderado]-----\n");
w4 = funciones(4, M, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w4, M, M)

%Dibujar pesos
figure();
c = categorical(["Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv Pond"]);
bar(c, [w1';w2';w3';w4']);
l=compose("A%d",(1:length(M)));
legend(l);

clear;

```

```

fprintf("-----Consistente Incompleta-----\n");
w = [0.4 0.3 0.2 0.1]
M= zeros(4);
for i = 1:4
    for j = 1:4
        M(i,j) = w(i)/w(j);
    end
end

%El experto no opina sobre la opcion 1
M(1,[3 4]) = 0;
M([3 4],1) = 0;

M
ic(M)

% M =
%
%      1      1.333      0      0
%      0.75      1.0000      1.5000      3.0000
%      0      0.6667      1.0000      2.0000
%      0      0.3333      0.5000      1.0000

fprintf("-----Metodo de la potencia-----\n");
w0 = funciones(0, M) % potencia(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w0, M)

fprintf("-----Metodo de Minimos Cuadrados [Log]-----\n");
w1 = funciones(1, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w1, M)

fprintf("-----Metodo de Minimos Cuadrados [Ponderado]-----\n");
w2 = funciones(2, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w2, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Log]-----\n");
w3 = funciones(3, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w3, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Ponderado]-----\n");
w4 = funciones(4, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w4, M)

%Dibujar pesos
figure();
c = categorical(["Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv Pond"]);
bar(c, [w1';w2';w3';w4']);
l=compose("A%d", (1:length(M)));
legend(l);

clear;

```

```

fprintf("-----NO Consistente-----\n");
M = [1.0000    0.1429    0.1429    0.2000;
      7.0000    1.0000    0.5000    0.3333;
      7.0000    2.0000    1.0000    0.1111;
      5.0000    3.0000    9.0000    1.0000];

ic(M)

fprintf("-----Metodo de la potencia-----\n");
w0 = funciones(0, M) % potencia(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w0, M)

fprintf("-----Metodo de Minimos Cuadrados [Log]-----\n");
w1 = funciones(1, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w1, M)
fprintf("-----Metodo de Minimos Cuadrados [Ponderado]-----\n");
w2 = funciones(2, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w2, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Log]-----\n");
w3 = funciones(3, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w3, M)

fprintf("-----Metodo de Minimo Suma Desviaciones [Ponderado]-----\n");
w4 = funciones(4, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w4, M)

%Dibujar pesos
figure();
c = categorical(["Potencia", "Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv
Pond"]);
bar(c, [w0';w1';w2';w3';w4']);
l=compose("A%d", (1:length(M)));
legend(l);

```

```

fprintf("-----NO Consistente Incompleta Varios Expertos-----\n");
M1 = [1    0    1/7    1/5;
      0    1    1/2    1/3;
      7    2     1    1/9;
      5    3     9     1];

M2 = [1  0  1/3 1/9;
      0  1   0 1/8;
      3  0   1 1/9;
      9  8   9  1];

M3 = [1 1/3 0 0;
      3  1  1/2 1/5;
      0  2   1  0;
      0  5   0  1];

%Tomamos M3 traspuesta
ic(M1)
ic(M2)
ic(M3')

fprintf("-----Metodo de Minimos Cuadrados [Log]-----\n");
w1 = funciones(1, M1, M2, M3') % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w1, M1, M2, M3')

fprintf("-----Metodo de Minimos Cuadrados [Ponderado]-----\n");
w2 = funciones(2, M1, M2, M3') % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w2, M1, M2, M3')

fprintf("-----Metodo de Minimo Suma Desviaciones [Log]-----\n");
w3 = funciones(3, M1, M2, M3') % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w3, M1, M2, M3')

fprintf("-----Metodo de Minimo Suma Desviaciones [Ponderado]-----\n");
w4 = funciones(4, M1, M2, M3') % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w4, M1, M2, M3')

%Dibujar pesos
figure();
c = categorical(["Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv Pond"]);
bar(c, [w1';w2';w3';w4']);
l=compose("A%d",(1:length(M)));
legend(l);

%Dibujar digrafo
grafo(M1, M2, M3')

```

funciones.m

```
function w = funciones(metodo, varargin)
% Juntar las matrices de los expertos
M = varargin{1};
for i = 2 : nargin-1
    M = [M; varargin{i}];
end

if nargin == 1
    [~, w, ~] = potencia(M);
else
    if metodo == 0 && nargin == 2
        [~, w, ~] = potencia(M);
    elseif metodo == 0 && nargin >= 2
        fprintf("Número erróneo de argumentos\n");
    elseif metodo == 1
        w = minCuadLog(varargin, nargin-1);
    elseif metodo == 2
        w = minCuadPond(varargin, nargin-1);
    elseif metodo == 3
        [w, ~, ~] = minSumDesvLog(varargin, nargin-1);
    else
        [w, ~, ~] = minSumDesvPond(varargin, nargin-1);
    end
end

return
```

potencia.m

```
function [lambda,x,iter] = potencia(A,tol,nmax,x0)
% Calcula el mayor (abs) autovalor lambda de A y un autovector asociado x
n=size(A,1);

if nargin == 1
    tol = 1e-06; % Tolerancia
    x0=rand(n,1); % Vector de arranque
    nmax=100; % N° máx iteraciones
end

x0=x0/norm(x0); x1=A*x0;
lambda=x0'*x1;
err = tol*abs(lambda) + 1;
iter=0;

while err > tol*abs(lambda) && abs(lambda) ~= 0 && iter <= nmax
    x=x1; x=x/norm(x);
    x1=A*x; lambda_new=x'*x1;
    err = abs(lambda_new - lambda);
    lambda=lambda_new;
    iter = iter + 1;
end

x = x / sum(x);
end
```

minCuadLog.m

```
function w = minCuadLog(E, n)

H = [];
b = [];

for i = 1:n
    M = E{i};

    s=size(M);
    long = (s(1)*s(2))-s(1);

    HAux = zeros([long,s(2)]);
    bAux = zeros([long,1]);

    k = 1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                notZero = (log(M(i,j)) ~= -Inf);
                % Generar H
                HAux(k,i) = notZero;
                HAux(k,j) = -1*notZero ;
                % Generar b
                if (notZero)
                    bAux(k) = log(M(i,j));
                else
                    bAux(k) = 0;
                end
                k = k+1;
            end
        end
    end
    H = [H; HAux];
    b = [b; bAux];
end

cond(H'*H)
% The result h is 1 if the test rejects the null hypothesis at the 5% significance level,
% that the data in vector x is from a population with a normal distribution,
% using the Anderson-Darling test, or 0 otherwise.
[h,p,adstat,cv] = adtest(exp(b))
% Resolver con minimos cuadrados
v=H\b;
% Deshacer el cambio de logaritmo.
w=exp(v)
% Normalizar
w=w/sum(w);
return
```

minCuadPond.m

```
function w = minCuadPond(E, n)

H = [];
b = [];

for i = 1:n
    M = E{i};

    s = size(M);
    long = (s(1)*s(2))-s(1);

    HAux = zeros([long,s(2)]);

    % Generar b
    bAux=zeros([long,1]);
    %bAux(long+1) = 1;
    b = [b; bAux];

    k = 1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                % Generar
                %(M(i,j) == 0) devuelve 1 o 0 si queremos descartar
                %información que el experto no ha dado
                HAux(k,i) = (M(i,j) ~= 0);
                HAux(k,j) = -M(i,j);
                k=k+1;
            end
        end
    end

    H = [H; HAux];
end
cond(H'*H)
H = [H; ones(1, s(1))];
b = [b; 1];

% Resolver con minimos cuadrados
w=H\b;

% Normalizar
w=w/sum(w);

return
```


minSumDesvLog.m

```
function [w, n, p] = minSumDesvLog(E, n)

Aeq = [];
beq = [];

s = zeros(1, 2);

for i = 1:n
    M = E{i};

    s = size(M);
    longM = (s(1)*s(2))-s(1);

    % No hay desigualdades
    A = [];
    b = [];

    AeqAux = zeros(longM, s(2));
    beqAux = zeros(longM, 1);

    k=1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                notZero = (log(M(i,j)) ~= -Inf);
                % Generar Aeq
                AeqAux(k,i) = notZero;
                AeqAux(k,j) = -1*notZero ;
                % Generar beq
                if (notZero)
                    beqAux(k) = log(M(i,j));
                else
                    beqAux(k) = 0;
                end
                k = k + 1;
            end
        end
    end

    Aeq = [Aeq; AeqAux];
    beq = [beq; beqAux];
end

long = size(Aeq, 1);

%Añadimos las metas a las ecuaciones
N = eye(long);
P = -1 * eye(long);

Aeq = [Aeq N P];

x = zeros(s(1) + long + long, 1);
f = [zeros(s(1),1); ones(long,1); ones(long,1)];
```

```

% Todos los elementos de x mayor o igual que 0
lb = zeros(length(x), 1);

ub = [];

x = linprog(f, A, b, Aeq, beq, lb, ub);

% Coger de x la parte que nos interesa
v = x(1:s(1));
% Deshacer el cambio logaritmico
w = exp(v);
% Normalizar
w = w/sum(w);

n = x(s(1)+1:s(1)+long);
p = x(s(1)+long+1:s(1)+2*long);
end

```

minSumDesvPond.m

```

function [w, n, p] = minSumDesvPond(E, n)

Aeq = [];
beq = [];

s = zeros(1, 2);

for i = 1:n
    M = E{i};

    s = size(M);
    longM = (s(1)*s(2))-s(1);

    % No hay desigualdades
    A = [];
    b = [];

    AeqAux = zeros(longM, s(2));

    k = 1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                % Generar Aeq
                AeqAux(k,i) = (M(i,j) ~= 0);
                AeqAux(k,j) = -M(i,j);
                k = k + 1;
            end
        end
    end

    Aeq = [Aeq; AeqAux];
end

long = size(Aeq, 1);

```

```

%Añadimos las metas a las ecuaciones
N = eye(long);
P = -1 * eye(long);
Aeq = [[Aeq N P]; [ones(1, s(1)) zeros(1, 2*long)]];

x = zeros(s(1) + long + long, 1);
f = [zeros(s(1),1); ones(long,1); ones(long,1)];

beq = zeros([long+1,1]);
beq(long+1) = 1;

% Todos los elementos de x mayor o igual que 0
lb = zeros(length(x), 1);

ub = [];

x = linprog(f, A, b, Aeq, beq, lb, ub);

% Coger de x la parte que nos interesa
w = x(1:s(1));

% Normalizar
w = w/sum(w);

n = x(s(1)+1:s(1)+long);
p = x(s(1)+long+1:s(1)+2*long);
end

```

errores.m

```

function [errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w,varargin)

datos_conocidos_total = 0;
errorInf = [];
IndexMaxErr = [];
errorFro = [];
errorUno = [];
errorNoAciertos = [];

M = [];
W = [];

% Construccion W
WAux = zeros(size(varargin{1}));
for i = 1:size(w)
    for j = 1:size(w)
        WAux(i,j) = w(i)/w(j);
    end
end
end

```

```

% Errores de cada experto
for k = 1:nargin

    if k == nargin
        E = M;
        WAux = W;
    else
        E = varargin{k};

        % Contar 0s de M
        datos_conocidos = 0;
        for m = 1:length(E)
            for n = 1:length(E)
                if E(m,n) ~= 0
                    datos_conocidos = datos_conocidos + 1;
                end
            end
        end
    end

    WAux2 = WAux .* zerosM(E);
    % Matriz de residuos
    R = abs(E-WAux2) ;

    % Norma Infinito: residuo maximo y su indice
    [errorInfAux, I] = max(R(:));
    errorInf = [errorInf errorInfAux/datos_conocidos];
    [I_row, I_col] = ind2sub(size(R),I);
    IndexMaxErr = [IndexMaxErr [I_row, I_col]];

    % Norma Frobenius
    errorFroAux = norm(R,'fro');
    errorFro = [errorFro errorFroAux/datos_conocidos];

    % Norma 1
    errorUnoAux = sum(sum(R));
    errorUno = [errorUno errorUnoAux/datos_conocidos];

    % Norma "ErrorRel"
    % errorErrRelAux = norm(R./M, 'fro'); Puede dividir por 0
    % errorErrRel = [errorErrRel errorErrRelAux/datos_conocidos];

    % Norma "No aciertos"
    [m, n] = size(E);
    errorNoAciertosAux = 0;
    for i = 1:m
        for j = 1:n
            if (E(i,j) > 0)
                if w(i) > w(j) && E(i,j) < 1
                    errorNoAciertosAux = errorNoAciertosAux + 1;
                elseif (w(i) == w(j) && E(i,j) ~= 1) || (w(i) ~= w(j) && E(i,j) == 1)
                    errorNoAciertosAux = errorNoAciertosAux + 0.5;
                end
            end
        end
    end
end
end

```

```

    errorNoAciertos = [errorNoAciertos errorNoAciertosAux/datos_conocidos];

    % Si solo nos pasan una matriz
    if (nargin == 2)
        break;
    end

    M = [M; E];
    W = [W; WAux2];
    datos_conocidos_total = datos_conocidos_total + datos_conocidos;

end

if nargin > 2
    figure();
    l=compose("E%d", (1:nargin-1));
    % Pie
    tiledlayout(1, 3);
    ax1 = nexttile;
    pie(ax1, errorInf(1:end-1)./sum(errorInf(1:end-1)))
    legend(l)
    title('Error Inf')

    ax2 = nexttile;
    pie(ax2,errorFro(1:end-1)./sum(errorFro(1:end-1)))
    legend(l)
    title('Error Frobenius')

    ax3 = nexttile;
    pie(ax3,errorUno(1:end-1)./sum(errorUno(1:end-1)))
    legend(l)
    title('Error 1')
end

end

```

ic.m

```

function res = ic(M)
%Autovalor dominante
lambda = max(eig(M));
m = length(M);
%Calculo del indice de consistencia
res = (lambda - m) / (m - 1);
return

```

zerosM.m

```

function mBin = zerosM(M)
    mBin=(M~=0);
return

```