Relación 2: AHP con R+ahp (Problemas 2.4 y 2.6)

Pedro Luis Luque

Contents

1 Pasos iniciales

```
source("teoriadecision_funciones_multicriterio.R")
source("teoriadecision_funciones_multicriterio_diagram.R")

library(formattable)
library(htmltools)
library(webshot)
export_formattable <- function(f, file, width = "100%", height = NULL,
                               background = "white", delay = 0.2)
{
  w <- formattable::as.htmlwidget(f, width = width, height = height)
  path <- htmltools::html_print(w, background = background, viewer = NULL)
  url <- paste0("file:///", gsub("\\\\", "/", normalizePath(path)))
  webshot::webshot(url,
          file = file,
          selector = ".formattable_widget",
          delay = delay)
}
```

2 Ejercicio 4

2.1 Con ayuda de las funciones R de clase

Se usan las funciones R en "teoriadecision_funciones_multicriterio.R":

- **Método 1**:
  - `multicriterio.crea.matrizvaloraciones_mej()`
  - `multicriterio.metodoAHP.variante1.autovectormayorautovalor()`
  - `multicriterio.metodoAHP.pesosglobales_entabla()`
- **Método 2**:
  - `multicriterio.crea.matrizvaloraciones_mej()`
  - `multicriterio.metodoAHP.variante3.completo()`

2.1.1 Método 1

```
tb0401 = multicriterio.crea.matrizvaloraciones_mej(c(2),numalternativas = 2,
            v.nombres.alternativas = c("Rendimiento","Riesgo"))
tb0401
```

```
##             Rendimiento Riesgo
## Rendimiento         1.0      2
## Riesgo              0.5      1
```

```
(tb0402a = multicriterio.crea.matrizvaloraciones_mej(c(3),2,c("A","B")))
```

```
##           A B
## A 1.0000000 3
## B 0.3333333 1
```

```
(tb0402b = multicriterio.crea.matrizvaloraciones_mej(c(1/2),2,c("A","B")))
```

```
##   A   B
## A 1 0.5
## B 2 1.0
```

Calculamos los pesos locales:

```
pl0401 = multicriterio.metodoAHP.variante1.autovectormayorautovalor(tb0401)
pl0401
```

```
## $Xmat
##             Rendimiento Riesgo
## Rendimiento         1.0      2
## Riesgo              0.5      1
##
## $autovalor
## [1] 2
##
## $suma.autovector
## [1] 1.341641
##
## $normaeuclidea.autovector
## [1] 1
##
## $valoraciones.ahp
## Rendimiento      Riesgo
##   0.6666667   0.3333333
```

```
##
## $valoraciones.ahp.ordenadas
## Rendimiento      Riesgo
##   0.6666667   0.3333333
##
## $CI.coef.inconsistencia
## [1] 0
##
## $RI.coef.inconsistencia
## [1] NaN
##
## $consistencia
## [1] "Consistencia aceptable"
##
## $tablaresumen
##             Rendimiento Riesgo    autovector.v prioridades.relativas
## Rendimiento         1.0      2  2    0.8944272             0.6666667
## Riesgo              0.5      1 NA    0.4472136             0.3333333
##                      NA     NA NA    1.3416408                    NA
```

```
pl0402a = multicriterio.metodoAHP.variante1.autovectormayorautovalor(tb0402a)
pl0402a$valoraciones.ahp
```

```
##    A    B
## 0.75 0.25
```

```
pl0402b = multicriterio.metodoAHP.variante1.autovectormayorautovalor(tb0402b)
pl0402b$valoraciones.ahp
```

```
##         A         B
## 0.3333333 0.6666667
```

Calculamos ahora los pesos globales:

```
pg04 = multicriterio.metodoAHP.pesosglobales_entabla(pl0401$valoraciones.ahp,
       rbind(pl0402a$valoraciones.ahp,
             pl0402b$valoraciones.ahp))
pg04
```

```
##                  Rendimiento    Riesgo Ponderadores Globales
## A                  0.7500000 0.3333333             0.6111111
## B                  0.2500000 0.6666667             0.3888889
## Ponder.Criterios   0.6666667 0.3333333                    NA
```

MEJOR DECISIÓN: ALTERNATIVA A (peso global del 61.11%)

2.1.2 Método 2

Con la función: `multicriterio.metodoAHP.variante3.completo()`

```
Xmat.criterios = multicriterio.crea.matrizvaloraciones(c(1,2,1/2,1),
                                                       2,c("Rendimiento","Riesgo"))
Xmat.rendimiento = multicriterio.crea.matrizvaloraciones(c(1,3,1/3,1),
                                                       2,c("Alt. A","Alt. B"))
Xmat.riesgo = multicriterio.crea.matrizvaloraciones(c(1,1/2,2,1),
                                                       2,c("Alt. A","Alt. B"))
num.alt = 2
num.cri = 2
```

```r
Xmatriznivel2 = array(NA,dim=c(num.alt,num.alt,num.cri))
Xmatriznivel2[,,1] = Xmat.rendimiento
Xmatriznivel2[,,2] = Xmat.riesgo
pg04com = multicriterio.metodoAHP.variante3.completo(Xmat.criterios,Xmatriznivel2)
#pg04com
pg04com$pesos.globales_entabla
```

```
##                  Rendimiento    Riesgo Ponderadores Globales
##                    0.7500000 0.3333333             0.6111111
##                    0.2500000 0.6666667             0.3888889
## Ponder.Criterios   0.6666667 0.3333333                    NA
```

MEJOR DECISIÓN: ALTERNATIVA A (peso global del 61.11%)

2.2 Con ayuda de las funciones R de clase (diagram)

Se usan las funciones R en "teoriadecision_funciones_multicriterio_diagram.R":

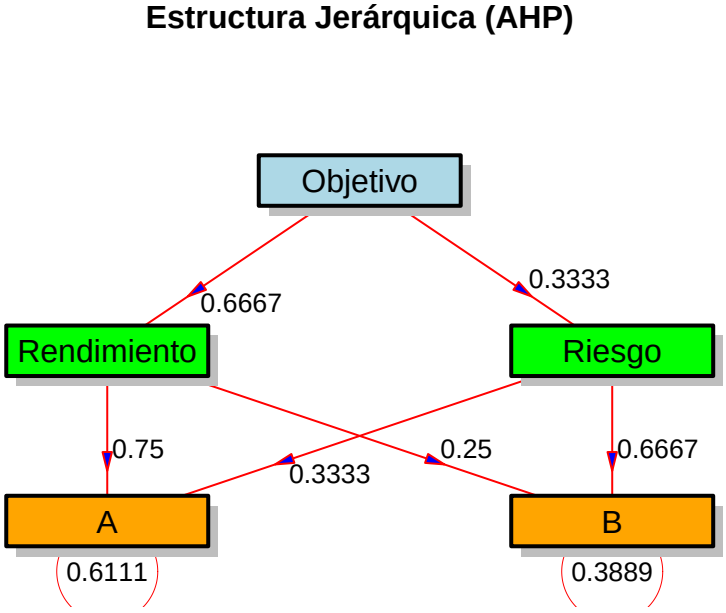
- **Método 1 (diagram)**:
  - `multicriterio.crea.matrizvaloraciones_mej()`
  - `multicriterio.metodoahp.diagrama()`

En forma de diagrama:

```
matriznivel2_04 = array(NA, dim = c(2,2,2))
matriznivel2_04[,,1] = tb0402a
matriznivel2_04[,,2] = tb0402b
dimnames(matriznivel2_04)[[1]] = c("A","B")
dimnames(matriznivel2_04)[[2]] = c("A","B")
matriznivel2_04
```

```
## , , 1
##
##           A B
## A 1.0000000 3
## B 0.3333333 1
##
## , , 2
##
##   A   B
## A 1 0.5
## B 2 1.0
```

```
# teoriadecision_funciones_multicriterio_diagram.R
multicriterio.metodoahp.diagrama(tb0401,matriznivel2_04)
```



MEJOR DECISIÓN: ALTERNATIVA A (peso global del 61.11%)

2.3 Uso del paquete "ahp"

Ver mi Fork: https://github.com/calote/ahp

```
#devtools::install_github("gluc/ahp", build_vignettes = TRUE)
devtools::install_github("calote/ahp", build_vignettes = TRUE)
```

Se usan las funciones R:

- **Método ahp**:
  - `datos = Load(fichero_ahp)`
  - `Calculate(datos)`
  - `Visualize(datos)`
  - `AnalyzeTable(datos, variable = "priority")` (pesos locales)
  - `AnalyzeTable(datos)` (contribuciones)
- Nota: uso de la función: `export_formattable( AnalyzeTable(datos), file = "fichero.png")`

Para utilizar el paquete R: "ahp" a través de la aplicación Shiny asociada.

```
library(ahp)
ahp::RunGUI()
```

Escribo el fichero "ahp24.ahp" con los datos. Y pasamos a resolverlo:

```
Version: 2.0
Alternatives: &alternatives
  A:
  B:
Goal:
  name: Aplicar AHP
  preferences:
    pairwise:
      - [Rendimiento, Riesgo, 2]
  children:
    Rendimiento:
      preferences:
        pairwise:
          - [A, B, 3]
      children: *alternatives
    Riesgo:
      preferences:
        pairwise:
          - [A, B, 1/2]
      children: *alternatives
```

```
library(ahp)
datos = Load("ahp24.ahp")
Calculate(datos)
```

```
Visualize(datos)
```

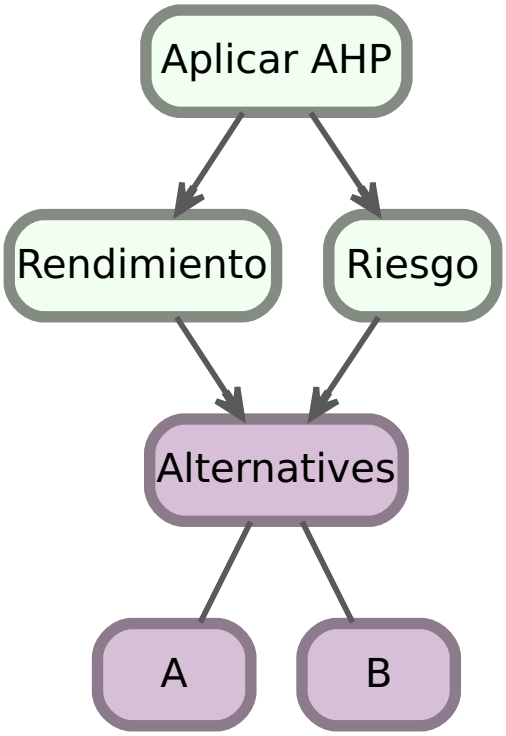


Tabla solución (contribución total)

```
export_formattable(AnalyzeTable(datos), file = "tablaahp204.png")
```

	Weight	A	B	Inconsistency
Aplicar AHP	100.0%	61.1%	38.9%	0.0%
Rendimiento	66.7%	50.0%	16.7%	0.0%
Riesgo	33.3%	11.1%	22.2%	0.0%

MEJOR DECISIÓN: ALTERNATIVA A (peso global del 61.1%)

Tabla solución (pesos locales)

```
t2 = AnalyzeTable(datos, variable = "priority")
export_formattable(t2, file = "tablaahp204b.png")
```

	Priority	A	B	Inconsistency
Aplicar AHP	100.0%			0.0%
Rendimiento	66.7%	75.0%	25.0%	0.0%
Riesgo	33.3%	33.3%	66.7%	0.0%

3 Ejercicio 6

3.1 Con ayuda de las funciones R en clase

3.1.1 Método 1

```
# teoriadecision_funciones_multicriterio.R
tb0601 = multicriterio.crea.matrizvaloraciones_mej(c(7,9,3),numalternativas = 3,
           v.nombres.alternativas = c("Costo","Confiabilidad","Plazo Entrega"))
tb0601
```

```
##                   Costo Confiabilidad Plazo Entrega
## Costo         1.0000000     7.0000000             9
## Confiabilidad 0.1428571     1.0000000             3
## Plazo Entrega 0.1111111     0.3333333             1
```

```
(tb0602a = multicriterio.crea.matrizvaloraciones_mej(c(1/3,6,8),3,c("A","B","C")))
```

```
##           A         B C
## A 1.0000000 0.3333333 6
## B 3.0000000 1.0000000 8
## C 0.1666667 0.1250000 1
```

```
(tb0602b = multicriterio.crea.matrizvaloraciones_mej(c(6,2,1/3),3,c("A","B","C")))
```

```
##           A B         C
## A 1.0000000 6 2.0000000
## B 0.1666667 1 0.3333333
## C 0.5000000 3 1.0000000
```

```
(tb0602c = multicriterio.crea.matrizvaloraciones_mej(c(8,1,1/8),3,c("A","B","C")))
```

```
##       A B     C
## A 1.000 8 1.000
## B 0.125 1 0.125
## C 1.000 8 1.000
```

Calculamos los pesos locales:

```
pl0601 = multicriterio.metodoAHP.variante1.autovectormayorautovalor(tb0601)
pl0601$valoraciones.ahp
```

```
##         Costo Confiabilidad Plazo Entrega
##    0.78539119    0.14881507    0.06579374
```

```
pl0602a = multicriterio.metodoAHP.variante1.autovectormayorautovalor(tb0602a)
pl0602a$valoraciones.ahp
```

```
##           A          B          C
## 0.28507705 0.65266352 0.06225944
```

```
pl0602b = multicriterio.metodoAHP.variante1.autovectormayorautovalor(tb0602b)
pl0602b$valoraciones.ahp
```

```
##   A   B   C
## 0.6 0.1 0.3
```

```
pl0602c = multicriterio.metodoAHP.variante1.autovectormayorautovalor(tb0602c)
pl0602c$valoraciones.ahp
```

```
##          A          B          C
## 0.47058824 0.05882353 0.47058824
```

Calculamos ahora los pesos globales:

```
pg06 = multicriterio.metodoAHP.pesosglobales_entabla(pl0601$valoraciones.ahp,
       rbind(pl0602a$valoraciones.ahp,
             pl0602b$valoraciones.ahp,
             pl0602c$valoraciones.ahp))
pg06
```

```
##                       Costo Confiabilidad Plazo Entrega Ponderadores Globales
## A                 0.28507705     0.6000000    0.47058824             0.3441478
## B                 0.65266352     0.1000000    0.05882353             0.5313479
## C                 0.06225944     0.3000000    0.47058824             0.1245043
## Ponder.Criterios 0.78539119     0.1488151    0.06579374                    NA
```

MEJOR DECISIÓN: ALTERNATIVA B (peso global del 53.1%)

3.1.2 Método 2

Con la función: `multicriterio.metodoAHP.variante3.completo()`

```
num.alt = 3
num.cri = 3
Xarray_nivel2 = array(NA,dim=c(num.alt,num.alt,num.cri))
Xarray_nivel2[,,1] = tb0602a
Xarray_nivel2[,,2] = tb0602b
Xarray_nivel2[,,3] = tb0602c
pg06com = multicriterio.metodoAHP.variante3.completo(tb0601,
                                                     Xarray_nivel2)
pg06com$pesos.globales_entabla
```

```
##                       Costo Confiabilidad Plazo Entrega Ponderadores Globales
##                  0.28952381     0.6000000    0.47058824             0.3500206
##                  0.64634921     0.1000000    0.05882353             0.5214694
##                  0.06412698     0.3000000    0.47058824             0.1285099
## Ponder.Criterios 0.77659202     0.1548978    0.06851022                    NA
```

MEJOR DECISIÓN: ALTERNATIVA B (peso global del 52.2%)

3.2 Estudio de la inconsistencia con funciones R de clase

Al ser matrices "3x3" hay que estudiar la inconsistencia

```
(Inconsistencia0601 = multicriterio.metodoAHP.coef.inconsistencia(tb0601))
```

```
## $lambda
## [1] 3.0803
##
## $m
## [1] 3
##
## $CI.coef.inconsistencia
## [1] 0.04014992
##
## $CA.aleatorio
## [1] 0.58
##
## $RI.coef.inconsistencia
## [1] 0.069224
##
## $mensaje
## [1] "Consistencia aceptable"
```

```
Inconsistencia0602a = multicriterio.metodoAHP.coef.inconsistencia(tb0602a)
c(Inconsistencia0602a$mensaje, round(Inconsistencia0602a$RI.coef.inconsistencia,4) )
```

```
## [1] "Consistencia aceptable" "0.0634"
```

```
Inconsistencia0602b = multicriterio.metodoAHP.coef.inconsistencia(tb0602b)
c(Inconsistencia0602b$mensaje, round(Inconsistencia0602b$RI.coef.inconsistencia,4) )
```

```
## [1] "Consistencia aceptable" "0"
```

```
Inconsistencia0602c = multicriterio.metodoAHP.coef.inconsistencia(tb0602c)
c(Inconsistencia0602c$mensaje, round(Inconsistencia0602c$RI.coef.inconsistencia,4) )
```

```
## [1] "Consistencia aceptable" "0"
```

3.3 Con ayuda de las funciones R en clase (diagram)

En forma de diagrama:

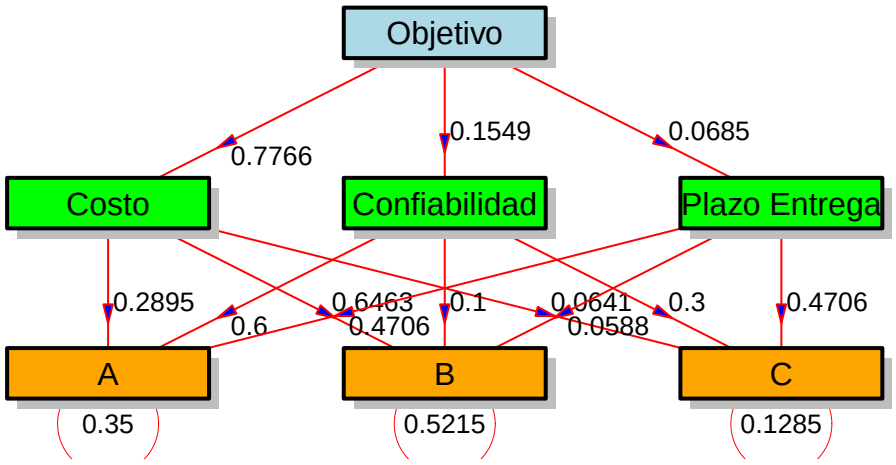
```
matriznivel2_06 = array(NA, dim = c(3,3,3))
matriznivel2_06[,,1] = tb0602a
matriznivel2_06[,,2] = tb0602b
matriznivel2_06[,,3] = tb0602c
dimnames(matriznivel2_06)[[1]] = c("A","B","C")
dimnames(matriznivel2_06)[[2]] = c("A","B","C")
matriznivel2_06
```

```
## , , 1
##
##           A         B C
## A 1.0000000 0.3333333 6
## B 3.0000000 1.0000000 8
## C 0.1666667 0.1250000 1
##
```

```
## , , 2
##
##           A B         C
## A 1.0000000 6 2.0000000
## B 0.1666667 1 0.3333333
## C 0.5000000 3 1.0000000
##
## , , 3
##
##       A B     C
## A 1.000 8 1.000
## B 0.125 1 0.125
## C 1.000 8 1.000
```

```
# teoriadecision_funciones_multicriterio_diagram.R
multicriterio.metodoahp.diagrama(tb0601,matriznivel2_06)
```

Estructura Jerárquica (AHP)



MEJOR DECISIÓN: ALTERNATIVA B (peso global del 52.2%)

3.4 Uso del paquete “ahp”

Para utilizar el paquete R: “ahp” a través de la aplicación Shiny asociada.

```
library(ahp)
ahp::RunGUI()
```

Escribo el fichero “ahp26.ahp” con los datos. Y pasamos a resolverlo:

```
Version: 2.0
Alternatives: &alternatives
  A:
  B:
  C:
Goal:
  name: Aplicar AHP
  preferences:
    pairwise:
      - [Costo, Confiabilidad, 7]
      - [Costo, Plazo Entrega, 9]
      - [Confiabilidad, Plazo Entrega, 3]
  children:
    Costo:
      preferences:
        pairwise:
          - [A, B, 1/3]
          - [A, C, 6]
          - [B, C, 8]
      children: *alternatives
    Confiabilidad:
      preferences:
        pairwise:
          - [A, B, 6]
          - [A, C, 2]
          - [B, C, 1/3]
      children: *alternatives
    Plazo Entrega:
      preferences:
        pairwise:
          - [A, B, 8]
          - [A, C, 1]
          - [B, C, 1/8]
      children: *alternatives
```

```
library(ahp)
datos = Load("ahp26.ahp")
Calculate(datos)
```

```
Visualize(datos)
```

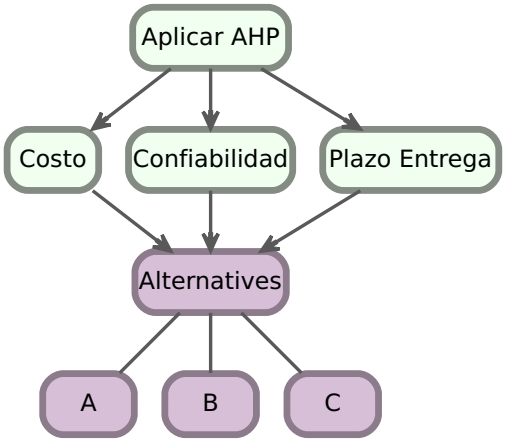


Tabla solución (contribución total)

```
export_formattable(AnalyzeTable(datos), file = "tablaahp206.png")
```

	Weight	B	A	C	Inconsistency
Aplicar AHP	100.0%	53.1%	34.4%	12.5%	7.7%
Costo	78.5%	51.3%	22.4%	4.9%	7.0%
Confiabilidad	14.9%	1.5%	8.9%	4.5%	0.0%
Plazo Entrega	6.6%	0.4%	3.1%	3.1%	0.0%

MEJOR DECISIÓN: ALTERNATIVA B (peso global del 53.1%)

Nota: coincide con la solución obtenida del **"Método 1 con variante1 para el cálculo de pesos locales"**

Tabla solución (pesos locales)

```
t2 = AnalyzeTable(datos, variable = "priority")
export_formattable(t2, file = "tablaahp206b.png")
```

	Priority	B	A	C	Inconsistency
Aplicar AHP	100.0%				7.7%
Costo	78.5%	65.3%	28.5%	6.2%	7.0%
Confiabilidad	14.9%	10.0%	60.0%	30.0%	0.0%
Plazo Entrega	6.6%	5.9%	47.1%	47.1%	0.0%