Resumen Parcial 1

Laura Mayorgas del Castillo

Contents

DECISIÓN AMBIENTE DE INCERTIDUMBRE

Previamenta antes de ejecutar cualquiera de las funciones se debe hacer llamar a los ficheros con las mismas, gracias al comando:

```
source("teoriadecision_funciones_incertidumbre.R")
```

Crear tabla de deción

Vemos un ejemplo con 4 estados de la naturaleza y 5 alternativas.

```
tabla=cbind(tabla <- data.frame(
  e1 = c(4,5,-3,7,8),
  e2 = c(5,-1,3,7,2),
  e3 = c(-4,9,8,2,1),
  e4 = c(4,6,7,-9,3)
))
row.names(tabla)=c("d1","d2","d3","d4","d5")
```

Criterio de Wald

```
Wald=criterio.Wald(tb01,favorable = TRUE)
```

Criterio Optimista

```
Optimista=criterio.Optimista(tb01,favorable = TRUE)
```

Criterio de Hurwitcz

```
Hurwitcz=criterio.Hurwicz(tb01,favorable = TRUE)
dibuja.criterio.Hurwicz(tb01,favorable = TRUE)
```

Criterio de Savage

```
Savage=criterio.Savage(tb01,favorable = TRUE)
```

Criterio Laplace

```
LaPlace=criterio.Laplace(tb01,favorable = TRUE)
```

Criterio del Punto Ideal

```
PuntoIdeal=criterio.PuntoIdeal(tb01,favorable = TRUE)
```

Criterios Todos

```
Todos=criterio.Todos(tb01,favorable = T, alfa = 0.5)
```

nota: Esto es siempre es desde el punto de vista favorable, en caso contrario, de estar en una matriz de costos cambiar el argumento *favorable=F*

DECISIONES MULTICRITERIO

Previamenta antes de ejecutar cualquiera de las funciones se debe hacer llamar a los ficheros con las mismas, gracias al comando:

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

Crear tablas de valoraciones

Meter los datos por **filas** y hacer uso de la función *multicriterio.crea.matrizvaloraciones*

```
tabla <- multicriterio.crea.matrizvaloraciones(c(1,0,1,
                                                 1,1,1,
                                                 0,0,1), numalternativas = 3)
```

Función utilidad maximales

Nos quedaremos con el resultado **mayor** y podremos ordenar las alternativas por orden de preferencia

```
maximal <- multicriterio.constfuncutilidad.maximales(tabla) #suma por filas
sort(maximal, decreasing = T) #ordenar las alternativas
```

Función utilidad borroso

Mismo razonamiento,nos quedaremos con el resultado **mayor** y podremos ordenar las alternativas por orden de preferencia

```
fuzzy <- multicriterio.constfuncutilidad.estructuraborrosa(tabla) #calculo del flujo neto
sort(fuzzy, decreasing = T) #ordenar alternativas
```

Método Nadir

Necesitamos previamente **homogeneizar** los datos de la tabla, es decir, que todos los criterios estén evaluados en la misma unidad. Para ello ponemos todo entre valores de 0-1. Este proceso lo hace directamente la función programada.*Ejemplo:*

```
nadirHomogeneizado <- round(multicriterio.homogeneizacion.nadir(tabla), 4)
```

También usamos *round* para redondear los decimales de la tabla final, el último número 4 es el número de decimales que queremos.

Método Promethee

Para cada una de los criterios necesitamos un valor de:

$$\overline{\delta} \quad \underline{\delta}$$

. Vamos a inspeccionar cada uno de los argumentos de la función.

```
Promethee <- round(multicriterio.homogeneizacion.promethee(tab03,
                           v.delta.min = c(30,3,4,20,100), #conjunto de los delta abajo
                           v.delta.max = c(120,12,10,60,400)), 4) # deltas arriba
```

Igual que antes usamos *round* para limitar el número de decimales.

Método AHP

Funciones de clase

Introducción de datos En este método se deben hacer distintas matrices, una primera que compare los distintos criterios entre sí.

matriz criterios

```
MCriterios <- multicriterio.crea.matrizvaloraciones_mej(c(2,1/4,5), #triangulo superior
                                      numalternativas = 3,
                                      v.nombres.alternativas = c("Criterio1","Criterio2","Criterio3"))
```

Para rellenar solo se introducen los valores de la **triangular superior**

Posteriormente se debe hacer un estudio de cada una de las alternativas frente a cada criterio de forma análoga, exactamente igual.

matriz criterio 1

```
#matriz criterio 1
Criterio1 <- multicriterio.crea.matrizvaloraciones_mej(c(3), numalternativas = 2,
                                                       v.nombres.alternativas = c("A","B"))

#matriz criterio 2
Criterio2 <- multicriterio.crea.matrizvaloraciones_mej(c(1/2), numalternativas = 2,
                                                       v.nombres.alternativas = c("A","B"))
#matriz criterio 3
Criterio3 <- multicriterio.crea.matrizvaloraciones_mej(c(4), numalternativas = 2,
                                                       v.nombres.alternativas = c("A","B"))
```

Cálculo pesos locales

```
PesosCriterios <- multicriterio.metodoAHP.variante1.autovectormayorautovalor(MCriterios)
PesosCriterio1 <- multicriterio.metodoAHP.variante1.autovectormayorautovalor(Criterio1)
PesosCriterio2 <- multicriterio.metodoAHP.variante1.autovectormayorautovalor(Criterio2)
PesosCriterio3 <- multicriterio.metodoAHP.variante1.autovectormayorautovalor(Criterio3)
```

Método mayor autovalor

```
PesosCriterios <- multicriterio.metodoAHP.variante2.mediageometrica(MCriterios)
PesosCriterio1 <- multicriterio.metodoAHP.variante2.mediageometrica(Criterio1)
PesosCriterio2 <- multicriterio.metodoAHP.variante2.mediageometrica(Criterio2)
PesosCriterio3 <- multicriterio.metodoAHP.variante2.mediageometrica(Criterio3)
```

Método de media geométrica

```
PesosCriterios <- multicriterio.metodoAHP.variante3.basico(MCriterios)
PesosCriterio1 <- multicriterio.metodoAHP.variante3.basico(Criterio1)
PesosCriterio2 <- multicriterio.metodoAHP.variante3.basico(Criterio2)
PesosCriterio3 <- multicriterio.metodoAHP.variante3.basico(Criterio3)
```

Método básico

```
PesosGlobales <- multicriterio.metodoAHP.pesosglobales_entabla(
  PesosCriterios$valoraciones.ahp,
                                                    rbind(PesosCriterio1$valoraciones.ahp,
                                                          PesosCriterio2$valoraciones.ahp,PesosCrite
```

Cálculo pesos globales Nos quedaremos con aquella alternativa que muestre ***mayor peso global***

```
num.alt <- 2 # número de alternativas
num.crt <- 3 #número de criterios
Xmatriznivel <- array(NA, dim = c(num.alt, num.alt, num.crt))
Xmatriznivel[,,1] <- Criterio1
Xmatriznivel[,,2] <- Criterio2
Xmatriznivel[,,3] <- Criterio3
dimnames(Xmatriznivel)[[1]] <- c("ALTERNATIVA A","ALTERNATIVA B")
multicriterio.metodoahp.diagrama(MCriterios, Xmatriznivel)
```

Diagrama Jerarquías

Paquete AHP

Para eusar este paquete debemos tener los datos recogidos en un fichero *.ahp* con la estructura correcta, ver un ejemplo para entenderlo.

```
library(ahp)
datosAHP = Load("datos.ahp")
Calculate(datosAHP)
Visualize(datosAHP) #Muestra el diagrama de jerarquías
TablaSolución = AnalyzeTable(datos04, variable = "priority")
export_formattable(AnalyzeTable(datosAHP), file = "tablaAHP.png") #Exportar en un archivo .png
```

Método ELECTRE

Necesitamos varios datos para resolver el problema mediante este método: - Pesos de los criterios - Nivel de concordancia (general para todos los criterios) α - Nivel de discordancia para cada criterio d

```
Electre <- multicriterio.metodoELECTRE_I(tabla,
                                         pesos.criterios = c(0.25,0.25,0.2,0.2,0.2),
                                         nivel.concordancia.minimo.alpha = 0.7,
                                         no.se.compensan = c(60, Inf, 4, Inf, Inf),
                                         que.alternativas = T)

qgraph::qgraph(Electre$relacion.dominante)
Electre$nucleo_aprox
```

Seguir iterando hasta que el resultado de *nucleo_aprox* sea solo una de las alternativas. Por ejemplo si en la primer aiteración nos saliese que hay un empate entre a_1 y a_2,volver a calcular la respuesta pero cambiando el argumento $que.alternativas = T$ en la función por ***que.alternativas = c(1,2)***. También debemos ajustar los niveles de α.

ELECTRE I

```
Electre1 <- func_ELECTRE_Completo(Electre)
Electre1$Grafo
```

Vemos una tabla en la que nos indican la relación donde los elementos de la columna i superan siempre a los de la columna j. Se escribe $a_i S a_j$.

Método PROMETHEE

Ejemplo ej8 Para este método necesitamos tener simepre los pesos estandarizados, es decir, la suma de estos debe ser 1. En caso de no ser así se puede estandarizar dividiendo por la suma total de los pesos:

Datos (importante)

```
tablaPromethee <- multicriterio.crea.matrizdecision(c(-80,90-6,-5.4,-8,5,
                                                      -65,58,-2,-9.7,-1,1,
                                                      -83,60,-4,-7.2,-4,7,
                                                      -40,80,-10,-7.5,-7,10,
                                                      -52,72,-6,-2.0,-3,8,
                                                      -94,96,-7-3.6,-5,6),
                                                    numalternativas=6,
                                                    numcriterios=6,
                                                    v.nombresalt=c('A1','A2','A3','A4','A5','A6'), #nombre alter
v.nombrescri=c('f1', 'f2','f3','f4', 'f5','f6')) #nombre  criterios


pesos.criterios = c(1/6,1/6,1/6,1/6,1/6,1/6) #Pesos, en este caso equiprobables


tablaParametros = matrix (c(2,10,1,0,
                            3,0,30,0,
                            5,0.5,5,0,
                            4,1,6,0,
                            1,0,1,0,
                            6,0,1,5),ncol=4,byrow=T)
```

Especial atención a ***tablaParametros***. En ella se especifican:

Table 1: solo se detallan los 2 primeros criterios, análogo para el resto

Criterio	TIPO ELECTRE	q	p	s
c1	2	10	1	0
c2	3	0	30	0

PROMETHEE I

```
tab.Pthee.i = multicriterio.metodo.promethee_i(tablaPromethee,pesos.criterios,tablaParametros)
tab.Pthee.i

require ("qgraph")
qgraph(tab.Pthee.i$tablarelacionsupera)
```

PROMETHEE II

```
tab.Pthee.ii = multicriterio.metodo.promethee_ii(tablaPromethee,pesos.criterios,tablaParametros)
tab.Pthee.ii

qgraph(tab.Pthee.ii$tablarelacionsupera)
```

PROMETHEE I (medias)

```
tab.Pthee.i_med = multicriterio.metodo.promethee_i_med(tablaPromethee,pesos.criterios,tablaParametros)
tab.Pthee.i_med

qgraph (tab.Pthee.i_med$tablarelacionsupera)
```

PROMETHEE II (medias)

```
tab.Pthee.ii_med = multicriterio.metodo.promethee_ii_med(tablaPromethee,pesos.criterios,tablaParametros)
tab.Pthee.ii_med

qgraph (tab.Pthee.ii_med$tablarelacionsupera)
```

```
order(tab.Pthee.ii_med$vflujos.netos,decreasing = T)
```

Ordenación final alternativas Mét. Promethee II (medias)

```
order(tab.Pthee.ii$vflujos.netos,decreasing = T)
order(tab.Pthee.ii_med$vflujos.netos,decreasing = T)
```

Comparativa Promethee II: sin medias y con medias

Resolución con Promethee Windows

```
res = multicriterio.metodo.promethee_windows(tablaPromethee, tablaParametros, pesos.criterios)
res = multicriterio.metodo.promethee_windows (tablaPromethee, tablaParametros, pesos.criterios,
fminmax = c("min", "max", "min", "min","min","max"))

res02 = multicriterio.metodo.promethee_windows_kableExtra(res)
res02$tabEscenario
res02$tabAcciones
rownames(res$Acciones)
```