La práctica del análisis de correspondencias

MICHAEL GREENACRE

Catedrático de Estadística en la Universidad Pompeu Fabra

Separata del apéndice B

Cálculo del análisis de correspondencias

Primera edición: julio 2008
ISBN: 978-84-96515-71-0

Traducción: Jordi Comas Angelet
Revisión: Carles M. Cuadras Avellana

www.fbbva.es

Apéndice B

Cálculo del análisis de correspondencias

En este apéndice veremos el cálculo del AC utilizando el lenguaje de programación R, un software de alto nivel que podemos bajar libremente de la página web:

http://www.r-project.org

Supondremos que el lector ya tiene algunos conocimientos básicos sobre este lenguaje, que se ha convertido de facto en el software estándar para el cálculo estadístico. En caso contrario, en el sitio de Internet mencionado podemos encontrar muchos recursos para aprenderlo. Los programas que veremos en este apéndice se hallan también en el web de la red CARME (siglas en inglés de *Correspondence Analysis and Related MEthods*):

http://www.carme-n.org

Al final de este apéndice veremos también algunos programas comerciales y describiremos diferentes opciones para la creación de mapas.

Contenido

El programa R

El programa R proporciona todas las herramientas necesarias para obtener mapas de AC. La más importante es la descomposición en valores singulares (DVS). Estas herramientas son las *funciones* R. Algunas funciones y material relacionado los encontramos en forma de *paquetes* R. Así, el paquete **ca** nos permite llevar a cabo todas las modalidades del AC que hemos descrito en este libro. Lo iremos

viendo en este apéndice. También veremos el paquete **rgl** que nos permite crear mapas en tres dimensiones. De todas formas, empezaremos paso a paso haciendo algunos cálculos sencillos utilizando R. Con la letra tipo Courier indicaremos las instrucciones y las salidas en R. Por ejemplo, vamos a crear la matriz (13.2) de la página 137. Calcularemos su DVS y la guardamos en un objeto R tipo «svd». Luego visualizaremos la parte del objeto etiquetada como «d» (los valores propios):

```
table.T <- matrix(c(8,5,-2,2,4,2,0,-3,3,6,2,3,3,-3,-6,
                    -6,-4,1,-1,-2),nrow=5)
table.SVD <- svd(table.T)
table.SVD$d
[1] 1.412505e+01 9.822577e+00 1.376116e-15 7.435554e-32
```

(Las instrucciones las expresamos en color marrón, y los datos y resultados en color verde.)

Entrada de datos en R

La entrada de datos en R tiene sus peculiaridades. Sin embargo, una vez dominadas éstas, ¡el resto es muy fácil! La funcion `read.table()` es muy útil para introducir matrices de datos. Las fuentes de datos más fácilmente manejables son los archivos de texto o los archivos Excel. Por ejemplo, supongamos que queremos introducir la tabla de datos de 5 × 3 sobre los tipos de lectura que mostramos en la tabla de la imagen 3.1. Veamos tres opciones para leer estos datos.

1. Supongamos que los datos se hallan en un archivo texto como el siguiente:

```
      C1   C2   C3
E1     5    7    2
E2    18   46   20
E3    19   29   39
E4    12   40   49
E5     3    7   16
```

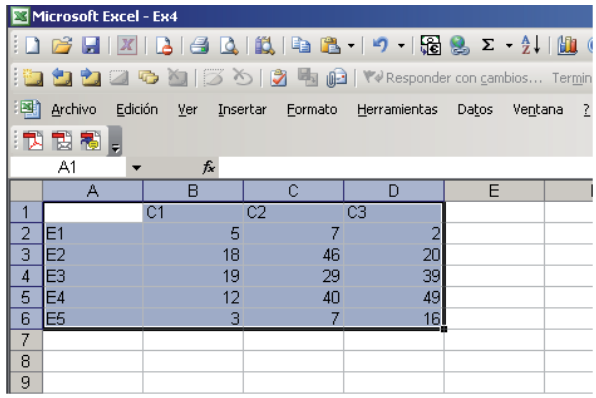
   que llamamos `reader.txt` y que hemos guardado en el directorio de trabajo R. Para leer los datos ejecutaremos la instrucción R siguiente:

```
read.table("reader.txt")
```

2. Otra posibilidad es seleccionar la tabla con el procesador de textos o con Word y luego copiarlo en el portapapeles utilizando la opción copiar del menú de Edit o clicando el botón de la derecha del ratón (suponiendo una plataforma Windows). Para leer directamente la tabla contenida en el portapapeles ejecutaremos la instrucción siguiente:

```
read.table("clipboard")
```

3. De manera similar, podemos leer los datos de un archivo Excel,* seleccionando los datos como mostramos a continuación:



y copiarlos en el portapapeles. A continuación ejecutaremos la siguiente instrucción:

```
table <- read.table("clipboard")
```

Con esta opción la tabla queda guardada como un *data frame* de R llamado `table`. Para indicar en la función `read.table()` que la primera línea contiene las etiquetas de las columnas y que la primera columna de las líneas posteriores contiene las etiquetas de las filas, tenemos que dejar un espacio vacío en la primera línea de la tabla copiada —así, podemos ver que hemos dejado una celda vacía en la esquina de arriba a la izquierda de la tabla Excel. Haríamos lo mismo si se tratara de un archivo de texto. Podemos ver el contenido de `table` ejecutando:

```
table
```

```
     C1   C2   C3
E1    5    7    2
E2   18   46   20
E3   19   29   39
E4   12   40   49
E5    3    7   16
```

El objeto incluye las etiquetas de filas y columnas. Las podemos ver escribiendo `rownames(table)` y `colnames(table)`, por ejemplo:

* Utilizando el paquete **foreign** de R (que se distribuye con el programa) es posible leer otros formatos, como por ejemplo Stata, Minitab, SPSS, SAS, Systat y DBF.

```
rownames(table)
[1] "E1" "E2" "E3" "E4" "E5"
```

Textos R para cada capítulo

A continuación describiremos de forma sistemática cómo realizar con R los cálculos de cada uno de los capítulos del libro. Empezaremos por el capítulo 2 viendo las funciones más básicas de R y el paquete **rgl** de representación gráfica tridimensional. Dejaremos para más adelante las explicaciones del paquete **ca**, que realiza los cálculos del AC de una manera mucho más compacta.

Capítulo 2: Perfiles y espacio de perfiles

En el capítulo 2 mostramos algunos diagramas triangulares correspondientes a los datos de mis viajes. Supongamos que hemos introducido los datos de los perfiles de la tabla de la imagen 2.1, como hemos descrito anteriormente, y que los guardamos en el *data frame* `profiles` de R:

```
profiles <- read.table("clipboard")
profiles
```

	Dias_de_fiesta	Medias_jornadas	Jornadas_completas
Noruega	0.333	0.056	0.611
Canada	0.067	0.200	0.733
Grecia	0.138	0.862	0.000
Francia/Alemania	0.083	0.083	0.833

(Fijémonos en que no hay espacios en blanco en las etiquetas, si los hubiera, los datos no se habrían leído correctamente.) Podemos generar una imagen en tres dimensiones de los perfiles utilizando el paquete **rgl**[*] de la manera siguiente (suponemos que hemos instalado y cargado **rgl**):

Ejemplo de figura tridimensional utilizando el paquete **rgl**

```
rgl.lines(c(0,1.2), c(0,0),   c(0,0))
rgl.lines(c(0,0),   c(0,1.2), c(0,0))
rgl.lines(c(0,0),   c(0,0),   c(0,1.2))
rgl.lines(c(0,0),   c(0,1),   c(1,0), size = 2)
rgl.lines(c(0,1),   c(1,0),   c(0,0), size = 2)
rgl.lines(c(0,1),   c(0,0),   c(1,0), size = 2)
rgl.points(profiles[,3], profiles[,1], profiles[,2], size = 4)
rgl.texts(profiles[,3], profiles[,1], profiles[,2],
    text = row.names(profiles))
```

En la figura de la imagen B.1 mostramos el diagrama de dispersión tridimensional desde un determinado punto de visión. Presionando el botón de la izquierda del ratón podemos hacer girar la figura para dar una mejor sensación tridimensional. En la figura de la imagen B.2 mostramos una de estas rotaciones en la que

[*] El paquete **rgl** no es uno de los paquetes proporcionados como estándar con R, hay que instalarlo bajándolo de la página web de R o de www.carme-n.org.

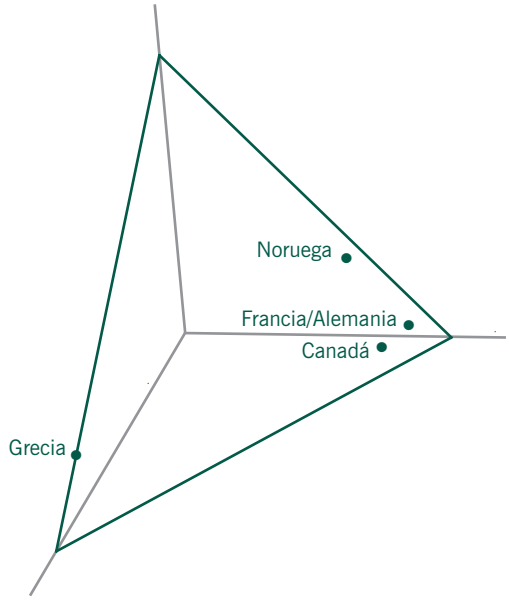


Imagen B.1:
Vista tridimensional de los perfiles fila de los países con los datos sobre los viajes, utilizando el paquete **rgl** *en* R

el punto de vista es plano con relación al triángulo que contiene los puntos correspondientes a los perfiles. La rueda del ratón permite acercar la imagen.

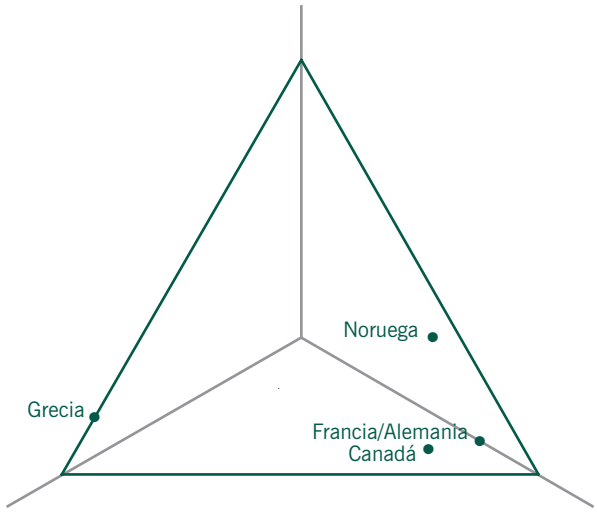
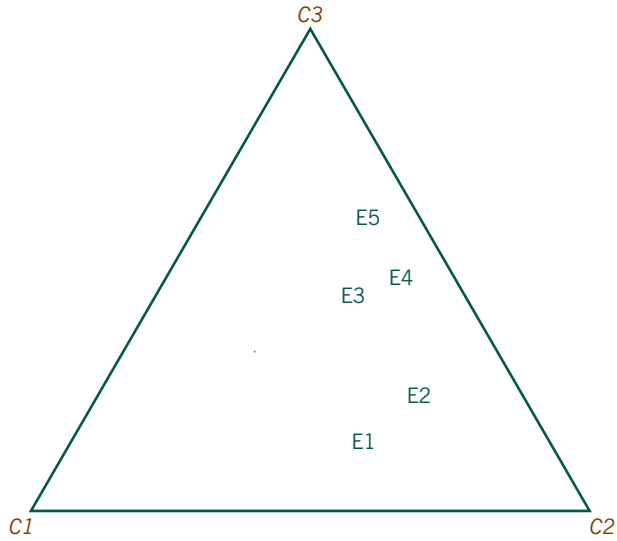


Imagen B.2:
Rotación del espacio tridimensional para mostrar dónde se hallan los puntos correspondientes a los perfiles

Como ilustración de las figuras del capítulo 3, vamos a ver cómo dibujar utilizando R, el diagrama de coordenadas triangular de la imagen 3.2. Para calcular las posiciones (x, y) de los puntos necesitamos un poco de trigonometría.

Capítulo 3:
Masas y centroide

Imagen B.3:
Diagrama de los perfiles de cinco niveles educativos en el espacio de coordenadas triangular



La función `apply()`

Supongamos que hemos leído la tabla como vimos al final de la página 267 y que tenemos los datos almacenados en el *data frame* `table`. Las siguientes instrucciones en R permiten obtener la figura de la imagen B.3. En la primera instrucción calculamos los perfiles de las filas mediante la función `apply()` y los guardamos en `table.pro`. Podemos aplicar la función `apply()` tanto a filas como a columnas. En nuestro caso el parámetro «`1`» indica filas, y `sum` la suma de sus elementos. En este caso dividimos los elementos de cada una de las columnas por la suma de los elementos de las correspondientes filas. En las dos líneas siguientes calculamos las coordenadas *x* e *y* de los cinco perfiles en un triángulo equilátero de lado 1. Para ello utilizamos los valores del primer y del tercer perfil (para situar los puntos sólo necesitamos dos de las tres coordenadas).

Ejemplo de figura bidimensional

```
table.pro <- table/apply(table, 1, sum)
table.x   <- 1 - table.pro[,1] - table.pro[,3]/2
table.y   <- table.pro[,3] * sqrt(3)/2
plot.new()
lines(c(0,1,0.5,0), c(0,0,sqrt(3)/2,0), col = "gray")
text(c(0,1,0.5), c(0,0,sqrt(3)/2), labels = colnames(table))
text(table.x, table.y, labels = rownames(table))
```

Capítulo 4: Distancias ji-cuadrado e inercia

En el capítulo 4 calculamos el estadístico χ^2, la inercia y las distancias χ^2. Vamos a ver cómo llevar a cabo cada uno de estos cálculos. Los realizaremos con los datos sobre los tipos de lectura que anteriormente guardamos en el *data frame* `table`.

— *Estadístico χ^2 e inercia total*

```
table.rowsum <- apply(table, 1, sum)
table.colsum <- apply(table, 2, sum)
table.sum    <- sum(table)
table.exp    <- tabla.rowsum %o% table.colsum / table.sum
chi2         <-sum((table - table.exp)^2 / table.exp)
chi2
[1] 25.97724
chi2 / table.sum
[1] 0.08326039
```

Producto externo, el operador `%o%`

Fijémonos en la utilización del operador `%o%` de *producto externo*, en la cuarta línea del programa anterior. Multiplica cada elemento del vector situado a su izquierda por cada elemento del vector situado a su derecha.

— *Distancias χ^2 de los perfiles fila al centroide*

Vamos a ver cómo calcular el cuadrado de la distancia χ^2 para la quinta fila de la tabla de perfiles, como vimos en (4.4). Para calcular la suma de los tres términos utilizamos la iteración `for` de R:

Ejemplo de iteración en R *utilizando* `for`

```
chidist <- 0
for(j in 1:3)
{chidist<-chidist + (table.pro[5,j] –
table.colmass[j]^2/table.colmass[j]}
chidist
       C1
0.1859165
```

R da la etiqueta `C1` al valor de `chidist`, probablemente porque es la primera columna de la iteración. Otra posibilidad es calcular las cinco distancias de una vez. Para ello, tenemos que restar a cada fila de la matriz de perfiles el correspondiente valor de la columna que contiene las masas de las filas, elevar estas diferencias al cuadrado, y luego dividir otra vez cada fila por las masas para, finalmente, sumar las filas. Debido a que en R, las matrices se guardan como columnas de vectores, las operaciones con filas son ligeramente más complicadas. Una posible solución es: primero transponer la matriz de perfiles, utilizando la función de transposición `t()` para a continuación sumar las columnas del objeto transpuesto (anteriormente filas) utilizando la función `apply()` con los parámetros que indican las suma de columnas `2,sum`:

Función de transposición `t()`

```
apply((t(table.pro)-table.colmass)^2/table.colmass,2,sum)
        E1            E2            E3            E4            E5
0.35335967    0.11702343    0.02739229    0.03943842    0.18591649
```

Función de distancia `dist()`

Adición de filas utilizando `rbind()`

La versátil función `sweep()`

Podemos calcular todas las distancias χ^2 entre perfiles y, en particular, entre éstos y su perfil media, mediante la función `dist()` que, por defecto, calcula la distancia euclídea matricial entre las filas de una matriz. Vamos a añadir la fila que contiene las masas de las columnas (perfil fila medio) a la matriz de perfiles utilizando la función `rbind()` (adición de filas) para formar la matriz de perfiles de 6×3, `tablec.pro`. A continuación dividiremos cada elemento del perfil por la correspondiente raíz cuadrada de la media. Una alternativa a la utilización de la operación de transposición es utilizar la versátil función `sweep()`, similar a `apply()` pero con más opciones. En la tercera línea del texto en R que mostramos a continuación las opciones de la función `sweep()` son `2` (operar en columnas), `sqrt(table.colmass)`(el vector utilizado para la operación) y `"/"` (división):

```
tablec.pro <- rbind(tablec.pro, table.colmass)
rownames(tablec.pro)[6] <- "ave"
dist(sweep(tablec.pro, 2, sqrt(table.colmass), FUN="/"))
```

```
            E1        E2        E3        E4        E5
E2   0.3737004
E3   0.6352512 0.4696153
E4   0.7919425 0.5065568 0.2591401
E5   1.0008054 0.7703644 0.3703568 0.2845283
ave  0.5944406 0.3420869 0.1655062 0.1985911 0.4311803
```

El resultado de la función `dist()` es un objeto que contiene una matriz triangular con todas las distancias entre los cinco perfiles. La última línea de la salida del programa —que hemos etiquetado como "`ave`" (en la segunda línea del programa y que corresponde al perfil fila medio añadido)— muestra las distancias χ^2 de los perfiles a su media.

Capítulo 5: Representación de distancias ji-cuadrado

En el capítulo 5 visualizamos las distancias χ^2 comprimiendo los ejes de coordenadas mediante factores inversamente proporcionales a las raíces cuadradas de sus correspondientes masas. Para hacer la representación gráfica seguiremos una secuencia de codificación similar a la que vimos anteriormente para el diagrama tridimensional del capítulo 2, con la excepción de que dividiremos cada coordenada por `sqrt(table.colmass)`. Un detalle importante para reproducir la figura de la imagen 5.2 es decidir qué elementos del perfil van en cada dimensión. Lo dejamos como ejercicio para el lector (de todas formas, en la página web que mencionamos podemos encontrar el texto del programa).

Capítulo 6: Reducción de la dimensionalidad

En el capítulo 6 consideramos el AC como un método que permite reducir dimensiones. Vamos a llevar a cabo nuestra primera descomposición en valores singulares (DVS). Para ello, en primer lugar, introducimos los datos sobre la autopercepción de la salud en `health`. A continuación, seguimos los pasos que vimos en la página 266 del apéndice A. Las pasos preliminares (A.1-A.3) son los siguientes:

```
health.P    <-health/sum(health)
health.r    <-apply(health.P, 1, sum)
health.c    <-apply(health.P, 2, sum)
health.Dr   <-diag(health.r)
health.Dc   <-diag(health.c)
health.Drmh <-diag(1/sqrt(health.r))
health.Dcmh <-diag(1/sqrt(health.c))
```

La función `diag()` *para obtener matrices diagonal*

Las dos últimas instrucciones anteriores crean $\mathbf{D}_r^{-\frac{1}{2}}$ y $\mathbf{D}_c^{-\frac{1}{2}}$, respectivamente. Posteriormente necesitaremos estas matrices de forma repetida (el nombre de objeto `mh` viene de "minus half"). Para poder llevar a cabo el producto de matrices (A.4), tenemos que transformar el *data frame* `health.P` en una matriz del entorno R. Efectuaremos el producto de matrices utilizando el operador `%*%`. Realizamos la DVS indicada en (A.5) con la función `svd()`.

Operador para la multiplicación de matrices `%*%`

```
health.P   <- as.matrix(health.P)
health.S   <- health.Drmh %*% (health.P-health.r %o% health.c)
                          %*% health.Dcmh
health.svd <- svd(health.S)
```

Ejemplo de DVS, la función `svd()`

Calculamos las coordenadas principales y estándares (`pc` y `sc`) como vimos en (A.6-A.9):

```
health.rsc <-health.Drmh %*% health.svd$u
health.csc <-health.Dcmh %*% health.svd$v
health.rpc <-health.rsc %*% diag(health.svd$d)
health.cpc <-health.csc %*% diag(health.svd$d)
```

¡Y esto es todo! Las 14 instrucciones en R anteriores constituyen el algoritmo de cálculo del AC; para calcular coordenadas del mapa de AC simplemente tenemos que sustituir `health` por cualquier otro objeto.

Por ejemplo, para ver los valores de las coordenadas principales de las filas en el primer eje escribiremos:

```
health.rpc[,1]
[1] -0.37107411 -0.32988430 -0.19895401  0.07091332  0.39551813 ...
```

(Fijémonos en que los signos están cambiados con relación al mapa de la imagen 6.3. Ello ocurre a menudo cuando utilizamos distintos softwares. Podemos cambiar sin problemas los signos de todas las coordenadas.)

En el capítulo 7 vimos las propiedades del escalado óptimo del AC. Por tanto, en este capítulo no realizamos cálculos complicados. Simplemente ilustramos el cálculo de la transformación de la escala óptima (7.5) utilizando las funciones R

Capítulo 7: Escalado óptimo

para el cálculo de valores máximos y mínimos. (Debido al cambio de signos de las coordenadas del primer eje, hemos invertido la escala. Es decir, la escala transformada va de 0 = muy buena a 100 = muy mala. Restando 100 a los valores resultantes obtenemos los valores de la tabla de la imagen 7.2.):

```
health.range <- max(health.csc[,1] – min(health.csc[,1])
health.scale <- (health.csc[,1] – min(health.csc[,1]))
    *100/health.range
health.scale
[1] 0.00000 18.86467 72.42164 98.97005 100.00000
```

Capítulo 8: Simetría entre el análisis de filas y el de columnas

En el capítulo 8 vimos más propiedades de los resultados del AC. La figura de la imagen 8.5 no la creamos utilizando R, la creamos directamente en LaTeX (véanse las descripciones sobre la composición gráfica al final de este apéndice). A continuación veremos cómo utilizando algunas instrucciones R, podemos ilustrar las propiedades de máxima correlación del AC. Por ejemplo, la ecuación (8.2) de la página 92. Así, podemos calcular la correlación entre los valores de edad y de salud en la primera dimensión, $\phi^{\mathsf{T}}\mathbf{P}\gamma$, donde ϕ y γ son las coordenadas estándares en la primera dimensión, y $\mathbf{P}$ es la matriz de correspondencias.

```
health.cor <- t(health.rsc[,1] %*% health.P %*% health.csc[,1]
health.cor^2

            [,1]
[,1]  0.1366031
```

El cuadrado de esta correlación es la primera inercia principal (el resultado anterior aparece como una matriz de 1 × 1, ya que resulta de una multiplicación de matrices). El cálculo anterior de las correlaciones queda justificado por el hecho de que las varianzas son 1. Vamos a ver la estandarización (A.12), por ejemplo, para las filas.

```
t(health.rsc[,1]) %*% health.Dr %*% health.rsc[,1]

     [,1]
[,1]    1
```

Capítulo 9: Representaciones bidimensionales

Contacto inicial con el paquete **ca**

En el capítulo 9 vimos la geometría de los mapas bidimensionales. Comparamos los mapas asimétricos con los simétricos. Vamos a aprovechar que el paquete **ca** contiene los datos sobre los fumadores para iniciarnos en su utilización. Una vez instalado y cargado el paquete **ca**, podemos obtener los mencionados datos ejecutando la instrucción:

```
data(smoke)
```

que nos proporciona el *data frame* `smoke`:

```
smoke
```

```
     none  light  medium  heavy
SM      4      2       3      2
JM      4      3       7      4
SE     25     10      12      4
JE     18     24      33     13
SC     10      6       7      2
```

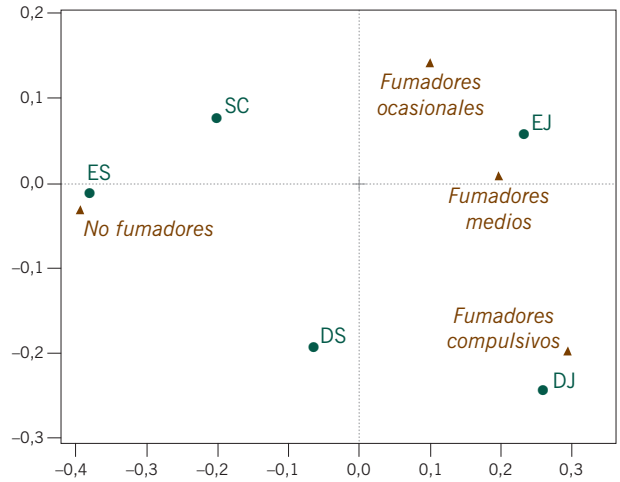
La función `ca()` —una de las funciones del paquete **ca**— nos permite llevar a cabo un AC simple. Así, podemos obtener fácilmente el AC de los datos sobre los fumadores escribiendo `ca(smoke)`:

```
ca(smoke)
```

```
Principal inertias (eigenvalues):

             1          2          3
Value        0.074759   0.010017   0.000414
Percentage   87.76%     11.76%     0.49%

Rows:

                SM         JM         SE         JE         SC
Mass      0.056995   0.093264   0.264249   0.455959   0.129534
ChiDist   0.216559   0.356921   0.380779   0.240025   0.216169
Inertia   0.002673   0.011881   0.038314   0.026269   0.006053
Dim. 1   -0.240539   0.947105  -1.391973   0.851989  -0.735456
Dim. 2   -1.935708  -2.430958  -0.106508   0.576944   0.788435

Columns:

              none      light     medium      heavy
Mass      0.316062   0.233161   0.321244   0.129534
ChiDist   0.394490   0.173996   0.198127   0.355109
Inertia   0.049186   0.007059   0.012610   0.016335
Dim. 1   -1.438471   0.363746   0.718017   1.074445
Dim. 2   -0.304659   1.409433   0.073528  -1.975960
```

De los resultados anteriores nos tendrían que resultar familiares las inercias principales y sus porcentajes. Y para filas y columnas, las masas, las distancias χ^2 al centroide, las inercias y las coordenadas estándares en las dos primeras dimensiones. Más adelante describiremos mucho más detalladamente las características de este paquete. Por el momento sólo vamos a mostrar lo fácil que resulta hacer una representación gráfica. Para obtener el mapa simétrico del AC de la imagen B.4 basta con escribir y ejecutar la función `plot()` con `ca(smoke)`:

Imagen B.4:
Mapa simétrico de los datos sobre los fumadores, utilizando el paquete **ca**




```
plot(ca(smoke))
```

Fijémonos en que, con relación al mapa de la imagen 9.5, aparecen invertidos los dos ejes principales. Para obtener mapas asimétricos, añadiremos la opción `map="rowprincipal"` o `map="colprincipal"` a la función `plot()`. Por ejemplo, podemos obtener el mapa de la imagen 9.2 de la siguiente manera:

```
plot(ca(smoke), map = "rowprincipal")
```

Capítulo 10: Tres ejemplos más

Lo que hemos visto sobre el paquete **ca** nos basta para poder llevar a cabo los análisis del capítulo 10. Los datos que utilizamos están disponibles en la página web www.carme-n.org en formatos texto y Excel. Los datos sobre los autores se hallan también en el paquete **ca**, de manera que los podemos obtener, igual que hicimos con los datos sobre los fumadores, con la instrucción R `data(author)`. Para visualizar los datos sobre los autores en un mapa de AC tridimensional, podemos probar lo siguiente (suponemos que hemos cargado el paquete **ca**):

```
data(author)
plot3d.ca(ca(author), labels = c(2,1), sf = 0.000001)
```

Capítulo 11: Contribuciones a la inercia

En el capítulo 11 introducimos algunos cálculos nuevos. Todos ellos implementados en el paquete **ca**. Sin embargo, igual que antes, primero hagamos los cálculos «a mano». Podemos leer los datos sobre la financiación de la investigación científica como describimos anteriormente —supongamos que hemos llamado `fund` al *data frame* que contiene estos datos. Calculamos la matriz de residuos estanda-

rizados de esta tabla como hicimos en el capítulo 4. Las inercias de la tabla de la imagen 11.1 son las sumas de cuadrados de las filas y de las columnas de la matriz de residuos.

```
fund.P    <- as.matrix(fund/sum(fund))
fund.r    <- apply(fund.P, 1, sum)
fund.c    <- apply(fund.P, 2, sum)
fund.Drmh <- diag(1/sqrt(fund.r))
fund.Dcmh <- diag(1/sqrt(fund.c))
fund.res  <- fund.Drmh %*% (fund.P - fund.r %o% fund.c) %*% fund.Dcmh
round(apply(fund.res^2, 1, sum), 5)
[1] 0.01135 0.00990 0.00172 0.01909 0.01621 0.01256 0.00083
[8] 0.00552 0.00102 0.00466
round(apply(fund.res^2, 2, sum),5)
[1] 0.01551 0.00911 0.00778 0.02877 0.02171
```

Contribuciones de cada celda de la tabla a la inercia total

Las contribuciones, expresadas en tantos por mil, de la tabla de la imagen 11.2 son los cuadrados de los residuos estandarizados con relación al total:

```
round (1000*fund.res^2/sum(fund.res^2), 0)
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	0	32	16	0	89
[2,]	0	23	4	44	48
[3,]	3	12	1	0	5
[4,]	9	15	11	189	8
[5,]	106	11	2	74	3
[6,]	1	11	38	1	102
[7,]	2	0	0	3	5
[8,]	51	4	0	10	2
[9,]	10	0	0	2	0
[10,]	5	3	22	26	0

(Después de las multiplicaciones de matrices, hemos perdido las etiquetas de filas y de columnas. Las podemos recuperar utilizando las funciones `rownames()` y `colnames()`.)

Las inercias principales de la tabla de la imagen 11.3 son los cuadrados de los valores singulares resultantes de hacer la DVS (`"svd"`) de la matriz de residuos:

```
fund.svd <- svd(fund.res)
fund.svd$d^2
[1] 3.911652e-02 3.038081e-02 1.086924e-02 2.512214e-03 4.252722e-33
```

(obtenemos cinco valores, sin embargo, el último es teóricamente igual a cero).

Para calcular los componentes individuales de la inercia de las filas en los cuatro ejes, en primer lugar, necesitamos calcular las coordenadas principales f_{ik} [véase (A.8)] y luego los valores $r_i f_{ik}^2$:

```
fund.F    <- fund.Drmh %*% fund.svd$u %*% diag(fund.svd$d)
fund.rowi <- diag(fund.r) %*% fund.F^2
fund.rowi[,1:4]
```

```
                [,1]          [,2]          [,3]          [,4]
 [1,]  6.233139e-04  9.775878e-03  8.222230e-04  1.301601e-04
 [2,]  1.178980e-03  7.542243e-03  8.385857e-04  3.423076e-04
 [3,]  2.314352e-04  8.787604e-04  2.931994e-04  3.211261e-04
 [4,]  1.615600e-02  1.577160e-03  6.274587e-04  7.271264e-04
 [5,]  1.426048e-02  1.043783e-04  1.691831e-03  1.562740e-04
 [6,]  1.526183e-03  9.407586e-03  1.273528e-03  3.573707e-04
 [7,]  7.575664e-06  5.589276e-04  7.980532e-05  1.868385e-04
 [8,]  3.449918e-03  1.601539e-04  1.799425e-03  1.091335e-04
 [9,]  5.659639e-04  7.306881e-06  4.185906e-04  3.022249e-05
[10,]  1.116674e-03  3.684113e-04  3.024590e-03  1.516545e-04
```

lo que concuerda con los valores de la tabla de la imagen 11.5. Fijémonos en que, en la última instrucción anterior, sólo hemos considerado las primeras cuatro columnas (`fund.rowi[,1:4]`). Dado que el quinto valor singular es cero, los valores de la quinta columna teóricamente son cero. Finalmente, expresamos estos componentes con relación a la inercia de un punto (suma de filas) o a la inercia de un eje (sumas de columnas, es decir, las inercias principales) [véanse (A.27) y (A.26), respectivamente], y al mismo tiempo los expresamos en tantos por mil, de la siguiente manera:

Cálculo de las contribuciones relativas (cosenos al cuadrado o correlaciones)

```
round(1000*(fund.rowi/apply(fund.rowi, 1, sum)) [,1:4], 0)
```

```
      [,1] [,2] [,3] [,4]
 [1,]   55  861   72   11
 [2,]  119  762   85   35
 [3,]  134  510  170  186
 [4,]  846   83   33   38
 [5,]  880    6  104   10
 [6,]  121  749  101   28
 [7,]    9  671   96  224
 [8,]  625   29  326   20
 [9,]  554    7  410   30
[10,]  240   79  649   33
```

lo que concuerda con los datos de la tabla de la imagen 11.6 (para obtener las calidades que aparecen en la tabla de la imagen 11.8 sumamos las primeras dos columnas de la tabla anterior). Respecto a las sumas de columnas, es decir, las inercias principales:

Cálculo de las contribuciones a cada eje principal

```
round(1000*t(t(fund.rowi)/fund.svd$d^2) [,1:4], 0)
```

```
      [,1] [,2] [,3] [,4]
 [1,]   16  322   76   52
 [2,]   30  248   77  136
 [3,]    6   29   27  128
 [4,]  413   52   58  289
 [5,]  365    3  156   62
 [6,]   39  310  117  142
 [7,]    0   18    7   74
 [8,]   88    5  166   43
 [9,]   14    0   39   12
[10,]   29   12  278   60
```

que muestra cómo se ha construido cada eje. Por ejemplo, las filas 4 y 5 (Física y Zoología) son las que más contribuyen al primer eje.

Anticipándonos un poco a la descripción completa del paquete **ca**, podemos ver que si aplicamos la función `summary()` a `ca(fund)` obtenemos los resultados completos del análisis anterior:

```
summary(ca(fund))
```

```
Principal inertias (eigenvalues):

      dim      value      %      cum%  scree plot
[1,]  1        0.039117   47.2   47.2  ************************
[2,]  2        0.030381   36.7   83.9  ******************
[3,]  3        0.010869   13.1   97.0  ******
[4,]  4        0.002512    3.0  100.0
[5,]           --------  -----
[6,]  Total:   0.082879  100.0

Rows:

     name   mass  qlt  inr     k=1  cor  ctr     k=2  cor  ctr
 1 |  Gel |  107  916  137 |    76   55   16 |   303  861  322 |
 2 |  Bic |   36  881  119 |   180  119   30 |  -455  762  248 |
 3 |  Chm |  163  644   21 |    38  134    6 |    73  510   29 |
 4 |  Zol |  151  929  230 |  -327  846  413 |   102   83   52 |
 5 |  Phy |  143  886  196 |   316  880  365 |    27    6    3 |
 6 |  Eng |  111  870  152 |  -117  121   39 |  -292  749  310 |
 7 |  Mcr |   46  680   10 |    13    9    0 |  -110  671   18 |
 8 |  Bot |  108  654   67 |  -179  625   88 |   -39   29    5 |
 9 |  Stt |   36  561   12 |   125  554   14 |    14    7    0 |
10 |  Mth |   98  319   56 |   107  240   29 |   -61   79   12 |
```

```
Columns:
     name   mass  qlt  inr     k=1  cor  ctr     k=2  cor  ctr
1  |    A |   39  587  187 |   478  574  228 |    72   13    7 |
2  |    B |  161  816  110 |   127  286   67 |   173  531  159 |
3  |    C |  389  465   94 |    83  341   68 |    50  124   32 |
4  |    D |  162  968  347 |  -390  859  632 |   139  109  103 |
5  |    E |  249  990  262 |   -32   12    6 |  -292  978  699 |
```

Capítulo 12: Puntos adicionales

En el capítulo 12 vimos cómo añadir puntos a un mapa utilizando la relación baricéntrica existente entre las coordenadas estándares de las columnas y las coordenadas principales de las filas. Es decir, los perfiles se hallan situados a medias ponderadas de los vértices. El ejemplo que vimos en la página 130 muestra cómo situar el punto adicional *Museos* [4 12 11 19 7], cuya suma total es 53. Si el vector **m** contiene el perfil de *Museos,* los productos escalares de éste con las coordenadas estándares de las columnas, $\mathbf{m}^{\mathsf{T}}\Gamma$, proporciona las coordenadas buscadas:

```
fund.m     <- c(4,12,11,19,7)/53
fund.Gamma <- fund.Dcmh %*% fund.svd$v
t(fund.m) %*% fund.Gamma[,1:2]
```

```
             [,1]       [,2]
[1,]  -0.3143203  0.3809511
```

(En comparación con el mapa de la imagen 12.2, en esta solución, el signo del segundo eje aparece cambiado. Si llevamos a cabo la misma operación con los vectores unitarios de la tabla de la imagen 12.4 como puntos adicionales, y luego los multiplicamos por las coordenadas estándares de las columnas, veríamos que sus posiciones coinciden con estas últimas.)

Capítulo 13: Biplots de análisis de correspondencias

En el capítulo 13 vimos las diferentes escalas de los biplots del AC. En el mapa correspondiente al biplot estándar del AC de la imagen 13.3, las filas están en coordenadas principales y las columnas en coordenadas estándares multiplicadas por la raíz cuadrada de las masas respectivas de las columnas. Dadas las coordenadas estándares calculadas anteriormente en `fund.Gamma`, el cálculo de las coordenadas en este biplot estándar, en las dos primeras dimensiones, es el siguiente:

```
diag(sqrt(fund.c)) %*% fund.Gamma[,1:2]
```

```
              [,1]         [,2]
[1,]   0.47707276   0.08183444
[2,]   0.25800640   0.39890356
[3,]   0.26032157   0.17838093
[4,]  -0.79472740   0.32170520
[5,]  -0.08046934  -0.83598151
```

Perfiles estimados a partir del biplot

En las siguientes instrucciones, en notación matricial, primero guardamos los productos escalares del lado derecho de (13.7), para $K^* = 2$, en `fund.est` y luego calculamos los perfiles estimados multiplicando por las raíces cuadradas $\sqrt{c_j}$ y sumando c_j:

```
fund.est <- fund.F[,1:2] %*% t(diag(sqrt(fund.c)) %*%
    fund.Gamma[,1:2])
oner <- rep(1, dim(fund)[1])
round(fund.est %*% diag(sqrt(fund.c)) + oner %o% fund.c, 3)
```

```
           A     B     C     D     E
 [1,]  0.051 0.217 0.436 0.177 0.120
 [2,]  0.049 0.107 0.368 0.046 0.431
 [3,]  0.044 0.176 0.404 0.160 0.217
 [4,]  0.010 0.143 0.348 0.280 0.219
 [5,]  0.069 0.198 0.444 0.065 0.225
 [6,]  0.023 0.102 0.338 0.162 0.375
 [7,]  0.038 0.145 0.379 0.144 0.294
 [8,]  0.021 0.136 0.356 0.214 0.272
 [9,]  0.051 0.176 0.411 0.124 0.238
[10,]  0.048 0.162 0.400 0.120 0.270
```

resultado que podemos comparar con los verdaderos valores de los perfiles:

```
round(fund.P/fund.r, 3)
```

```
           A     B     C     D     E
Geol   0.035 0.224 0.459 0.165 0.118
Bioc   0.034 0.069 0.448 0.034 0.414
Chem   0.046 0.192 0.377 0.162 0.223
Zool   0.025 0.125 0.342 0.292 0.217
Phys   0.088 0.193 0.412 0.079 0.228
Engi   0.034 0.125 0.284 0.170 0.386
Micr   0.027 0.162 0.378 0.135 0.297
Bota   0.000 0.140 0.395 0.198 0.267
Stat   0.069 0.172 0.379 0.138 0.241
Math   0.026 0.141 0.474 0.103 0.256
```

Calculando las diferencias entre los valores verdaderos y los valores estimados de los perfiles obtenemos una aproximación a los errores individuales. La suma de los cuadrados de estas diferencias, convenientemente ponderadas, nos da un error general del AC bidimensional. Tenemos que ponderar cada fila de diferencias al cuadrado con la correspondiente masa de la fila r_i y cada columna con la inversa del valor esperado $1/c_j$. El cálculo es el siguiente (se trata de una instrucción empaquetada en dos líneas, ¡un ejemplo de programación R concentrada!):

```
sum(diag(fund.r) %*% (fund.est %*% diag(sqrt(fund.c))+
    oner %o% fund.c - fund.P/fund.r)^2 %*% diag(1/fund.c))
[1] 0.01338145
```

Para ver que el resultado es correcto, tenemos que sumar las inercias principales pero *no* las de los dos primeros ejes:

```
sum(fund.svd$d[3:4]^2)
[1] 0.01338145
```

lo que confirma los cálculos anteriores (es el 16% de la inercia no explicada que aparece en la página 143).

Calibración de los ejes del biplot

El cálculo de las calibraciones de los biplots es bastante complicado ya que implica mucha trigonometría. En vez de dar un listado de todo el procedimiento, recomendamos a los lectores interesados que consulten la página web en la que detallamos la programación de la función `biplot.ca()` que calcula las coordenadas de los puntos inicial y final, así como todas las marcas de los ejes del biplot para las columnas.

Capítulo 14: Relaciones de transición y de regresión

En el capítulo 14 vimos varias relaciones lineales entre las coordenadas de filas, de columnas y de datos. Aquí ilustraremos algunas de estas relaciones utilizando la función de modelización lineal de R, `lm()`, que permite especificar pesos en la regresión de mínimos cuadrados. Por ejemplo, vamos a llevar a cabo la regresión de mínimos cuadrados de las coordenadas estándares de las filas (eje *y* de la figura de la imagen 14.2) con relación a las coordenadas estándares (eje *x*). Las variables de la regresión tienen 10 × 5 valores que podemos vectorizar, partiendo de la matriz original expresada en columnas. De esta manera, la variable *x* es el vector (que llamaremos `fund.vecc`) en el que las coordenadas de la primera columna en la primera dimensión se repiten 10 veces, luego la segunda coordenada 10 veces, y así sucesivamente. Mientras que la variable *y* (`fund.vecr`) tiene repetidas las coordenadas de la primera dimensión cinco veces en una columna (calculamos las coordenadas estándares de las filas como `fund.Phi`). Cuando llevemos a cabo los cálculos podemos comprobar los valores de `fund.vecc` y de `fund.vecr`. Los pesos de las regresiones serán las frecuencias de la tabla original `fund`; para vectorizarlos, tenemos que convertir el *data frame* primero en una matriz y luego en un vector utilizando `as.vector()`:

Conversión de objetos de datos utilizando `as.matrix()` *y* `as.vector()`

```
fund.vec  <- as.vector(as.matrix(fund))
fund.Phi  <- fund.Drmh %*% fund.svd$u
fund.vecr <- rep(fund.Phi[,1], 5)
fund.vecc <- as.vector(oner %*% t(fund.Gamma[,1]))
```

Llevamos a cabo la regresión de mínimos cuadrados ponderada de la manera siguiente:

Ejemplo de `lm()`*, función para regresiones lineales utilizando la opción* `weights`

```
lm(fund.vecr~fund.vecc, weights = fund.vec)

Call:
lm(formula = fund.vecr ~ fund.vecc, weights = fund.vec)

Coefficients:
(Intercept)     fund.vecc
 -2.015e-16     1.978e-01
```

lo que muestra que la constante es cero y que el coeficiente es 0,1978, la raíz cuadrada de la primera inercia principal.

Para llevar a cabo la regresión descrita en la página 151 entre los cocientes de contingencia de Geología con relación a las coordenadas estándares en las primeras dos dimensiones, llevamos a cabo la regresión de la respuesta `fund.y` sobre las dos los primeras columnas de la matriz de coordenadas Γ en `fund.Gamma`, con los pesos $\mathbf{c}$ en `fund.c`, de la manera siguiente (para obtener más resultados, aplicamos la función `summary()` a la instrucción `lm()`):

```
fund.y <- (fund.P[1,]/fund.r[1]/fund.c
summary (lm(fund.y ~ fund.Gamma[,1] + fund.Gamma[,2],
    weights = fund.c))

Call:
lm(formula = fund.y ~ fund.Gamma[, 1]+fund.Gamma[, 2], weights = fund.c)

Residuals:
       A          B          C          D          E
-0.079708   0.016013   0.037308  -0.030048  -0.003764

Coefficients:
                  Estimate  Std. Error    t value    Pr(>|t|)
(Intercept)        1.00000     0.06678     14.975     0.00443 **
fund.Gamma[, 1]    0.07640     0.06678      1.144     0.37105
fund.Gamma[, 2]    0.30257     0.06678      4.531     0.04542 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06678 on 2 degrees of freedom
Multiple R-Squared: 0.9161, Adjusted R-squared: 0.8322
F-statistic: 10.92 on 2 and 2 DF, p-value: 0.0839
```

resultado que confirma los coeficientes que vimos al final de la página 151 (otra vez, el segundo coeficiente tiene el signo opuesto debido a que las coordenadas de la segunda dimensión tienen también signos opuestos) y R^2 es 0,916.

La función `lm()` no proporciona los coeficientes de regresión estandarizados. Sin embargo los podemos obtener utilizando la función de covarianza ponderada `cov.wt()` con la opción `cor=TRUE` para el cálculo de correlaciones ponderadas.

Ejemplo de la función `cov.wt()` *para calcular la correlación ponderada*

```
cov.wt(cbind(fund.y,fund.Gamma[,1:2], wt = fund.c, cor = TRUE)$cor $cor

            [,1]          [,2]          [,3]
[1,]  1.0000000  2.343286e-01  9.280040e-01
[2,]  0.2343286  1.000000e+00  2.359224e-16
[3,]  0.9280040  2.359224e-16  1.000000e+00
```

lo que concuerda, excepto por algunos cambios de signo, con la matriz de correlaciones que vimos al principio de la página 152.

Capítulo 15: Agrupación de filas o de columnas

En el capítulo 15 vimos la agrupación de Ward para realizar la agrupación ponderada de filas o de columnas con sus masas. La función R `hclust()` con la que llevamos a cabo la agrupación jerárquica no permite ponderar [véase (15.2)]. Tampoco lo permite la función `agnes()` del paquete **cluster**. Sin embargo, el paquete estadístico comercial XLSTAT, que describiremos más adelante, sí presenta esta posibilidad que también incluye los programas R de Fionn Murtagh (pág. 327).

Capítulo 16: Tablas de múltiples entradas

En el capítulo 16, describimos la codificación interactiva de variables. Para llevarla a cabo partimos de los datos originales o bien de una tabla de múltiples entradas derivada de los mismos. Por ejemplo, en el caso de los datos sobre la salud que vimos en el capítulo 16, los datos originales tenían el siguiente aspecto (mostramos las primeras cuatro filas de un total de 6371):

```
. . .   health  age  gender    . . .
. . .      4     5      2      . . .
. . .      2     3      1      . . .
. . .      2     4      1      . . .
. . .      3     5      1      . . .
. . .      .     .      .      . . .
. . .      .     .      .      . . .
```

Para obtener la tabla de la imagen 16.2, tenemos que combinar las siete categorías de edad y las dos categorías de género, para formar una nueva variable `age_gender` con 14 categorías. Lo conseguimos con la siguiente transformación:

Codificación interactiva

```
age_gender <- 7*(gender – 1) + age
```

que numerará los grupos de edad de hombres (`gender=1`), de 1 a 7, y los de las mujeres (`gender=2`), de 8 a 14. A partir de ahí, crearemos la tabla de contingen-

cia cruzando las variables `age_gender` y `health`. En R, creamos las tablas de contingencia con la función `table()`. Por ejemplo,

Tablas de contingencia con `table()`

```
table(age-gender, health)
```

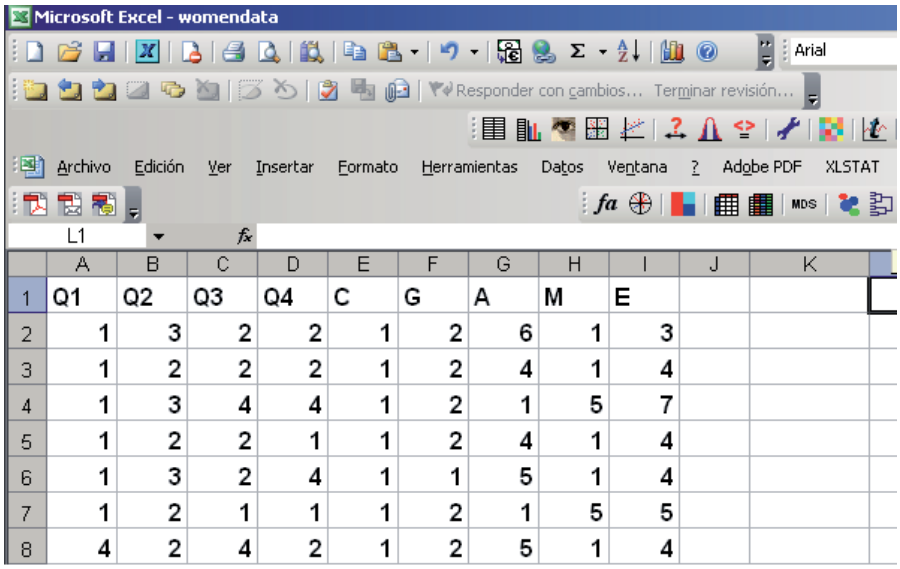
proporcionaría la tabla de contingencia de la imagen 16.2.

Supongamos ahora que los datos originales sobre el trabajo de las mujeres se halla en un archivo Excel como el que mostramos más adelante: cuatro preguntas, de `Q1` a `Q4`, país (`C`, de *country*), género (`G`, de *gender*), edad (`A`, de *age*), estado civil (`M`, de *marital status*) y educación (`E`, de *education*). Para introducir los datos en R, copiaremos, como vimos anteriormente las columnas en el portapapeles, y utilizaremos la función `read.table()`. Sin embargo, ahora las filas de la tabla no tienen nombres. Además, no hay un espacio en blanco en la celda de arriba a la izquierda. Por tanto, tenemos que especificar la opción `header=T` (`T` es la abreviatura de `TRUE`):

```
women <- read.table("clipboard"), header = T)
```

Podemos asignar nombres de las columnas del *data frame* `women` utilizando la función `colnames()`:

```
colnames(women)
[1] "Q1" "Q2" "Q3" "Q4" "C" "G" "A" "M" "E"
```



	A	B	C	D	E	F	G	H	I	J	K
1	Q1	Q2	Q3	Q4	C	G	A	M	E		
2	1	3	2	2	1	2	6	1	3		
3	1	2	2	2	1	2	4	1	4		
4	1	3	4	4	1	2	1	5	7		
5	1	2	2	1	1	2	4	1	4		
6	1	3	2	4	1	1	5	1	4		
7	1	2	1	1	1	2	1	5	5		
8	4	2	4	2	1	2	5	1	4		

Ejemplo de la función `attach()`

Para obtener la tabla de la imagen 16.4 podemos utilizar la función `attach()`, que permite disponer de las etiquetas de las columnas como si fueran un objeto de R (para que no sean disponibles podemos invertir la operación utilizando `detach()`):

```
attach(women)
table(C, Q3)
```

```
   Q3
C        1      2     3     4
   1   256   1156   176   191
   2   101   1394   581   248
   3   278    691    62    66
   4   161    646    70   107
   .     .      .     .     .
   .     .      .     .     .
  21   243    448   484    25
  22   468    664    92    63
  23   203    671   313   120
  24   738   1012   514   230
```

(compárese con la imagen 16.4).

Para obtener la tabla de la imagen 16.6 original con la variable fila codificada interactivamente:

```
CG <- 2*(C-1) + G
table(CG, Q3)
```

```
   Q3
CG       1     2     3     4
   1   117   596   114    82
   2   138   559    60   109
   3    43   675   357   123
   4    58   719   224   125
   .     .     .     .     .
   .     .     .     .     .
  47   348   445   294   112
  48   390   566   218   118
  51     1     2     0     0
  55     1     1     2     1
```

Fijémonos en que las dos últimas filas de la tabla corresponden a unos pocos valores perdidos de género que codificamos como `9`. Para visualizar los recuentos de frecuencias de las columnas, ejecutaremos la instrucción `lapply(women, table)`. De todas formas, primero deberíamos eliminar todos los valores perdidos —en la página 308 mostramos cómo eliminar las filas con valores perdidos—. También podríamos asignar el código NA de R a los valores perdidos. Así, para los valores perdidos de género de la columna 6:

```
women[,6] [G==9] <- NA
attach(women)
CG <- 2*(C – 1) + G
```

(fijémonos en que tenemos que aplicar de nuevo la función `attach()` al *data frame* `women` y recalcular `CG`). Suponiendo que hemos recodificado todos los valores perdidos (o eliminado las correspondientes filas), para construir una variable con 228 categorías que codifique interactivamente país, género, y edad (para edad no hay valores perdidos), codificamos las combinaciones de `CG` y `A` de la manera siguiente:

```
CGA <- 6*(CG – 1) + A
```

Capítulo 17: Tablas concatenadas

En el capítulo 17 vimos el AC de varias tablas de contingencia concatenadas. Las funciones `rbin()` y `cbin()` permiten agregar filas y columnas. Por ejemplo, supongamos que disponemos de la matriz de datos `women` sobre la que hemos aplicado la función `attached()` como vimos anteriormente. Podemos obtener la matriz compuesta de cinco tablas de contingencia correspondientes a la pregunta 3, que esquematizamos en la imagen 17.1 utilizando la iteración `for` de la siguiente manera:

```
women.stack <- table(C, Q3)
for (j in 6:9) {women.stack <- rbind(women.stack,
    table(women[,j], Q3))}
```

Podemos acceder a las columnas de `women` por su etiqueta o por el número de columna. Si miramos el contenido de `women.stack` veremos que para todas las variables demográficas, excepto país y grupo de edad, existen varias filas con valores perdidos. Antes de llevar a cabo el AC tenemos que omitir estos valores. Lo podemos hacer de tres maneras distintas: 1) excluyendo estas filas de la matriz; por ejemplo, podemos eliminar las filas 38, 39, 47 y 48 de la siguiente manera:

```
women.stack <- women.stack[-c(38,39,47,48),]
```

(el signo negativo antes de los números de las filas indica exclusión); 2) cambiando los códigos de los valores perdidos a NA, como describimos en la página anterior; o 3) declarando que las filas con valores perdidos se hallan fuera del subgrupo de interés en el AC de subgrupos que vimos en el capítulo 21 (dado que mantiene el tamaño de la muestra de todas las tablas, es la mejor opción).

El estadístico χ^2 utilizando la función de la prueba χ^2, `chisq.test()`

Para comprobar las inercias de la tabla de la página 168, podemos utilizar la función de la prueba χ^2 de R, `chisq.test()`. Uno de sus resultados es el estadístico χ^2, que podemos especificar mediante `$statistic`. Vamos a hacer los cálculos para la tabla de contingencia que cruza la variable edad con la pregunta 3, lo

que corresponde a las filas de la 27 a la 32 de la matriz compuesta (después de las 24 filas de país y las 2 de género). Obtenemos la inercia dividiendo el estadístico por el tamaño de la muestra, el total de la tabla.

```
chisq.test(women.stack[27:32,])$statistic/sum(women.stack[27:32,])
```

```
x-squared
0.0421549
```

Lo que concuerda con el valor de edad de la tabla de la página 168.

Para unir horizontalmente, con relación a las cuatro preguntas, las cuatro tablas (compuestas asimismo de cinco tablas unidas verticalmente) esquematizadas en la imagen 17.3 utilizaremos la función `cbind()` que permite unir columnas.

El paquete **ca**

Para llevar a cabo el ACM y métodos relacionados dejaremos los cálculos «a mano» que hemos utilizado hasta ahora, para empezar a utilizar las funciones del paquete **ca**. El paquete contiene funciones que permiten realizar el AC simple, múltiple y conjunto, así como funciones que facilitan el análisis de subgrupos y la inclusión de variables adicionales. También ofrece funciones para la representación gráfica de los resultados en dos y en tres dimensiones. El paquete comprende los siguientes componentes:

- AC simple
  - Cálculo: `ca()`
  - Salidas y resúmenes: `print.ca()` y `summary.ca()` (y `print.summary.ca()`)
  - Diagramas: `plot.ca()` y `plot3d.ca()`
- ACM y ACCo
  - Cálculo: `mjca()`
  - Salidas y resúmenes: `print.mjca()` y `summary.mjca()` (y `print.summary.mjca()`)
  - Diagramas: `plot.mjca()` y `plot3d.mjca()`
- Conjuntos de datos
  - `smoke`, `autor` y `wg93`

El paquete contiene más funciones, como `iterate.mjca()` para la actualización de la matriz de Burt en ACCo.

Función `ca()`

La función `ca()` calcula el CA simple, por ejemplo:

```
library(ca)    # carga el paquete ca, en caso de que no se haya hecho
                 antes utilizando el menú de R
```

```
data(smoke)
ca(smoke)
```

lleva a cabo un AC simple con los datos de `smoke` (véanse páginas 288-290). Con la función `names()` podemos obtener una lista de todos los componentes de `ca()`:

```
names(ca(smoke))
 [1]  "sv"          "nd"          "rownames"   "rowmass"    "rowdist"
 [6]  "rowinertia"  "rowcoord"    "rowsup"     "colnames"   "colmass"
[11]  "coldist"     "colinertia"  "colcoord"   "colsup"     "call"
```

Los resultados de `ca()` están estructurados como una lista de objetos. Por ejemplo, obtenemos las coordenadas estándares de las filas con:

```
ca(smoke)$rowcoord
```

La función `ca()` incluye una opción para fijar el número de dimensiones de la solución (`nd`), también incluye una opción para indicar las filas y/o columnas que queremos tratar como puntos adicionales (`suprow` y `supcol`, respectivamente) y opciones para indicar las filas y/o columnas que queremos seleccionar para llevar a cabo el AC de subgrupos (`subsetrow` y `subsetcol`, respectivamente). La función `summary()` permite obtener una salida más detallada:

```
summary(ca(smoke))
```

proporciona el siguiente resumen del AC:

```
Principal inertias (eigenvalues):

      dim      value      %      cum%  scree plot
[1,]  1        0.074759   87.8   87.8  ************************
[2,]  2        0.010017   11.8   99.5  ***
[3,]  3        0.000414    0.5  100.0
[4,]           --------  -----
[5,]  Total:   0.085190  100.0

Rows:

     name   mass   qlt  inr      k=1  cor  ctr      k=2  cor  ctr
1  |   SM |   57   893   31 |   -66   92    3 |  -194  800  214 |
2  |   JM |   93   991  139 |   259  526   84 |  -243  465  551 |
3  |   SE |  264  1000  450 |  -381  999  512 |   -11    1    3 |
4  |   JE |  456  1000  308 |   233  942  331 |    58   58  152 |
5  |   SC |  130   999   71 |  -201  865   70 |    79  133   81 |
```

```
Columns:
     name   mass   qlt  inr      k=1  cor  ctr      k=2  cor  ctr
1  |  non |  316  1000  577 |  -393  994  654 |   -30    6   29 |
2  |  lgh |  233   984   83 |    99  327   31 |   141  657  463 |
3  |  mdm |  321   983  148 |   196  982  166 |     7    1    2 |
4  |  hvy |  130   995  192 |   294  684  150 |  -198  310  506 |
```

Vemos que proporciona los valores propios y los porcentajes de inercia explicada de cada dimensión. También proporciona la inercia explicada en forma de porcentajes acumulados y el diagrama de descomposición (*scree plot* en inglés). En `Rows` y `Columns` encontramos las coordenadas principales de las dos primeras dimensiones (`k=1` y `k=2`). Junto con las coordenadas de los puntos hallamos las correlaciones al cuadrado (`cor`) y las contribuciones (`ctr`). Los valores de estas tablas están multiplicados por 1000. Por tanto, `cor` y `ctr` están expresadas en tantos por mil (‰). También proporciona la calidad (`qlt`) del resultado del AC solicitado. Así, en este ejemplo, la calidad es la suma de los cuadrados de las correlaciones de las dos primeras dimensiones. En el caso de haber variables adicionales, éstas se señalan con un asterisco junto a los nombres de las variables. Por ejemplo, el `summary` del AC de los datos `smoke`, en la que hemos considerado la categoría `none` (la primera columna) como una variable adicional, obtenemos:

```
summary(ca(smoke, supcol=1))
```

y en la correspondiente sección de la salida aparece lo siguiente:

```
...
Columns:
       name    mass qlt   inr     k=1 cor   ctr     k=2 cor   ctr
1 |  (*)non |  <NA>  55  <NA> |  292  39  <NA> |  -187  16  <NA> |
...
```

mostrando que las masas, las inercias y las contribuciones son *"not applicable"*.

Representaciones gráficas con el paquete **ca**

Por defecto, la función `plot()` del paquete **ca** visualiza los resultados del AC y del ACM en forma de mapas *simétricos* (`map="symmetric"`). Las restantes opciones son:

— `"symmetric"` Filas y columnas en coordenadas principales (por defecto), es decir, se realiza el calibrado de manera que la inercia es igual a la inercia principal (valor propio o cuadrado del valor singular).

— `"rowprincipal"` Filas en coordenadas principales y columnas en coordenadas estándares.

— `"colprincipal"` Columnas en coordenadas principales y filas en coordenadas estándares.

— `"symbiplot"` Las coordenadas de filas y columnas se calibran para que tengan inercias iguales a los valores singulares.

— `"rowgab"` Filas en coordenadas principales y columnas en coordenadas estándares multiplicadas por la masa (de acuerdo con la propuesta de Gabriel).

— `"colgab"` Columnas en coordenadas principales y filas en coordenadas estándares multiplicadas por la masa.

— `"rowgreen"` Filas en coordenadas principales y columnas en coordenadas estándares multiplicadas por la raíz cuadrada de la masa (de acuerdo con la propuesta de Greenacre [véase cap. 13]).

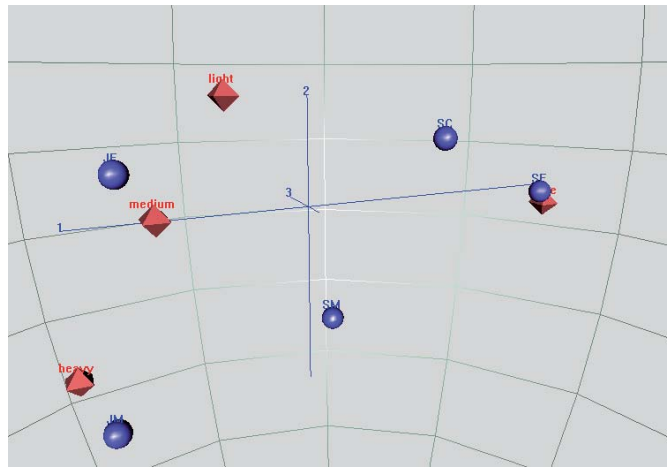
— `"colgreen"` Columnas en coordenadas principales y filas en coordenadas estándares multiplicadas por la raíz cuadrada de la masa.

Por defecto, las variables adicionales aparecen en el mapa con un símbolo distinto. Podemos definir los símbolos con la opción `pch` de `plot.ca()`. Esta opción toma cuatro valores en el orden siguiente: tipo de punto o símbolo para: 1) filas activas, 2) filas adicionales, 3) columnas activas y 4) columnas adicionales. Como regla general, las opciones que incluyen especificaciones para filas y para columnas contienen primero las de las filas y luego las de las columnas. Por ejemplo, especificamos el color de los símbolos con la opción de `col`. Por defecto, `col=c("#000000", "#FF0000")` (negro para las filas y rojo para las columnas). Además de estos códigos hexadecimales existe una lista reducida de nombres: `"black"`, `"red"`, `"blue"`, `"green"`, `"gray"`, etc.

Con la opción `what` podemos especificar las filas y las columnas que queremos visualizar en el mapa. Así podemos indicar `"all"` (todas), `"active"` (activas), `"passive"` (adicionales) o `"none"` (ninguna). Así, por ejemplo, con `what=c("active", "active")` creamos un diagrama con sólo puntos activos (es decir, sin puntos adicionales).

Además de las opciones de escalado de `map`, existen varias opciones que permiten añadir determinados atributos gráficos en los mapas. La opción `mass` hace que el tamaño de los puntos sea proporcional a su masa. De forma similar, utilizando la opción `contrib` podemos indicar mediante la intensidad de color del diagrama las contribuciones relativas o absolutas de los puntos.

Imagen B.5:
Mapa tridimensional de AC simple (lo podemos comparar con el mapa bidimensional de la imagen B.4)



La opción `dim` selecciona las dimensiones del mapa. Por defecto, `dim=c(1,2)` —es decir, se representan las primeras dos dimensiones—. Especificando `dim=c(2,3)`, obtendríamos un mapa con la segunda y la tercera dimensión. Para obtener un mapa en tres dimensiones podemos utilizar las funciones `plot3d.ca()` y `plot3d.mjca()`. Estas dos funciones necesitan del paquete **rgl** de R. Su estructura es similar a la de sus homólogos para dos dimensiones. Por ejemplo,

```
plot3d(ca(smoke, nd=3))
```

crea un mapa tridimensional de AC, como el que mostramos en la imagen B.5, que podemos hacer girar, aumentar o disminuir utilizando el ratón.

La función `mjca()` *del paquete* **ca**

Para llevar a cabo el ACM y el ACCo utilizamos la función `mjca()`. La estructura de esta función es similar a la del AC simple. Las dos diferencias más destacables son el formato de los datos de entrada y la limitación al análisis de columnas (sólo se proporcionan resultados para las columnas). Además, los puntos adicionales se limitan a columnas. Para ejecutar la función `mjca()` es necesario proporcionar los datos en forma de matriz; según el tipo de análisis que realicemos, la función transforma la matriz de datos en una matriz binaria o en una matriz de Burt. Podemos especificar el tipo de análisis a realizar con la opción `lambda` de la función `mjca()`:

— `lambda="indicator"`: análisis basado en un AC simple sobre la matriz binaria;

— `lambda="Burt"`: análisis basado en la descomposición en valores propios de la matriz de Burt;

— `lambda="adjusted"`: análisis basado en la matriz de Burt con inercias ajustadas (por defecto);

— `lambda="JCA"`: análisis de correspondencias conjunto.

Por defecto, la función `mjca()` lleva a cabo un análisis ajustado, es decir, `lambda="adjusted"`. En el ACC (`lambda="JCA"`), la matriz de Burt se actualiza iterativamente por mínimos cuadrados, mediante la función interna `iterate.mjca()`. Esta función de actualización tiene dos criterios de convergencia, `epsilon` y `maxit`. La opción `epsilon` compara la diferencia máxima absoluta de la matriz de Burt de cada iteración con la de la anterior. En la opción `maxit` especificamos el número máximo de iteraciones a realizar. El programa va iterando hasta que se satisface una de las dos condiciones anteriores. Podemos ignorar uno de los dos criterios indicando `NA`. Por ejemplo, podemos llevar a cabo exactamente 50 iteraciones, e ignorar el criterio de convergencia indicando `maxit=50` y `epsilon=NA`.

Igual que en el AC simple, mediante la opción `nd` podemos limitar la solución a dos dimensiones. Sin embargo, para las versiones «binaria» y «Burt» del ACM el programa proporciona los valores propios de $(J - Q)$ dimensiones. En el caso de un análisis ajustado o de un ACC, el programa proporciona sólo los valores propios de las k dimensiones, para las que los valores singulares de la matriz de Burt λ_k (es decir, las inercias principales de la matriz binaria) satisfacen la condición $\lambda_k > 1/Q$.

Capítulo 18: Análisis de correspondencias múltiples

En el capítulo 18, analizamos los datos sobre el trabajo de las mujeres, para las muestras de Alemania Occidental y del Este, utilizando la versión binaria y de Burt del ACM. Supongamos que previamente hemos leído el *data frame* `women` (con 33590 filas) y que hemos aplicado la función `attached()`. Los códigos de las dos muestras alemanas son 2 y 3, respectivamente. Podemos acceder a la parte de `women` correspondiente a estas dos muestras utilizando un vector *lógico* que llamaremos `germany`:

Ejemplo de operación lógica

```
germany <- C==2 / C==3
womenG <- women[germany,]
```

La primera instrucción crea un vector de longitud 33590 con valores `TRUE` para las filas de las muestras alemanas. En caso contrario, dichos valores son `FALSE`. La segunda instrucción crea un *data frame* llamado `womenG` sólo con las 3421 filas con valores `TRUE`. Sin embargo, la matriz que analizamos en el capítulo 18 tenía sólo 3418 porque eliminamos tres casos a los que les faltaban algunos datos demográficos. Supongamos que codificamos los valores perdidos de las variables género y estado civil con el código `9`, y los de educación con los códigos `98` y `99`.

Para eliminar las filas a las que les faltan valores seguimos los mismos pasos que vimos anteriormente, es decir, primero identificamos las filas con valores perdidos y luego las eliminamos:

Eliminación de valores perdidos fila a fila

```
missing <- G==9 / M ==9 / E ==98 / E ==99
womenG  <- (womenG[!missing,])
```

(si codificáramos los valores perdidos mediante el código `NA` de R, como vimos en la página 300, entonces para identificar y luego eliminar las correspondientes filas utilizaríamos este código).

Obtendremos la versión binaria del ACM correspondiente a las cuatro primeras columnas (las cuatro preguntas sobre el trabajo de las mujeres) de la siguiente manera:

```
mjca(womenG[,1:4], lambda = "indicator")

Eigenvalues:

            1        2        3        4        5        6
Value       0.693361 0.513203 0.364697 0.307406 0.21761 0.181521
Percentage  23.11%   17.11%   12.16%   10.25%   7.25%   6.05%
            7        8        9        10       11       12
Value       0.164774 0.142999 0.136322 0.113656 0.100483 0.063969
Percentage  5.49%    4.77%    4.54%    3.79%    3.35%    2.13%

Columns:

             Q1.1      Q1.2      Q1.3      Q1.4      Q2.1      Q2.2
Mass     0.182929  0.034816  0.005778  0.026477  0.013239  0.095012 ...
ChiDist  0.605519  2.486096  6.501217  2.905510  4.228945  1.277206 ...
Inertia  0.067071  0.215184  0.244222  0.223523  0.236761  0.154988 ...
Dim. 1  -0.355941 -0.244454 -0.279167  2.841498 -0.696550 -0.428535 ...
Dim. 2  -0.402501  1.565682  3.971577 -0.144653 -2.116572 -0.800930 ...
```

y la versión de Burt del ACM:

```
mjca(womenG[,1:4], lambda = "Burt")

Eigenvalues:

            1        2        3        4        5        6
Value       0.480749 0.263377 0.133004 0.094498 0.047354 0.03295
Percentage  41.98%   23%      11.61%   8.25%    4.13%    2.88%
            7        8        9        10       11       12
Value       0.027151 0.020449 0.018584 0.012918 0.010097 0.004092
Percentage  2.37%    1.79%    1.62%    1.13%    0.88%    0.36%
```

```
Columns:
             Q1.1      Q1.2      Q1.3      Q1.4      Q2.1      Q2.2
Mass     0.182929  0.034816  0.005778  0.026477  0.013239  0.095012 ...
ChiDist  0.374189  1.356308  3.632489  2.051660  2.354042  0.721971 ...
Inertia  0.025613  0.064046  0.076244  0.111452  0.073363  0.049524 ...
Dim. 1   0.355941  0.244454  0.279167 -2.841498  0.696550  0.428535 ...
Dim. 2  -0.402501  1.565682  3.971577 -0.144653 -2.116572 -0.800930 ...
```

En ambos casos, igual que en el AC simple, podemos calcular la inercia total como la suma de los cuadrados de los valores singulares:

```
sum(mjca(womenG[,1:4], lambda = "indicator")$sv^2)
[1] 3
sum(mjca(womenG[,1:4], lambda = "Burt")$sv^2)
[1] 1.145222
```

Mediante el componente `subinertia` de la función `mjca()` podemos obtener las contribuciones de cada una de las tablas de la matriz de Burt a la inercia total. A partir de su suma obtenemos la inercia total:

```
sum(mjca(womenG[,1:4], lambda = "Burt")$subinertia)
[1] 1.145222
```

Dado que la inercia total es la media de las inercias de las 16 tablas, la inercia de las tablas individuales es 16 veces los valores de `$subinertia`:

```
16*mjca(womenG[,1:4], lambda = "Burt")$subinertia
          [,1]      [,2]      [,3]      [,4]
[1,] 3.0000000 0.3657367 0.4261892 0.6457493
[2,] 0.3657367 3.0000000 0.8941517 0.3476508
[3,] 0.4261892 0.8941517 3.0000000 0.4822995
[4,] 0.6457493 0.3476508 0.4822995 3.0000000
```

Para hallar las posiciones de las variables adicionales:

```
summary(mjca(womenG, lambda = "Burt", supcol = 5:9))

Principal inertias (eigenvalues):

dim      value      %      cum%  scree plot
1        0.480749   42.0   42.0  *************************
2        0.263377   23.0   65.0  **************
3        0.133004   11.6   76.6  *******
4        0.094498    8.3   84.8  *****
5        0.047354    4.1   89.0  **
```

```
6         0.032950     2.9    91.9   **
7         0.027151     2.4    94.2   *
8         0.020449     1.8    96.0   *
9         0.018584     1.6    97.6   *
10        0.012918     1.1    98.8
11        0.010097     0.9    99.6
12        0.004092     0.4   100.0
          --------   -----
Total:    1.145222   100.0
```

```
Columns:
```

```
       name   mass  qlt  inr      k=1  cor  ctr      k=2  cor  ctr
1   |   Q1.1 |  183  740    6 |    247  435   23 |   -207  305   30 |
2   |   Q1.2 |   35  367   14 |    169   16    2 |    804  351   85 |
3   |   Q1.3 |    6  318   16 |    194    3    0 |   2038  315   91 |
4   |   Q1.4 |   26  923   24 |  -1970  922  214 |    -74    1    1 |
5   |   Q2.1 |   13  255   16 |    483   42    6 |  -1086  213   59 |
6   |   Q2.2 |   95  494   11 |    297  169   17 |   -411  324   61 |
.          .     .    .    .       .    .    .       .    .    .
.          .     .    .    .       .    .    .       .    .    .
17  | (*)C.2 | <NA>  283 <NA> |    -89   48 <NA> |    195  234 <NA> |
18  | (*)C.3 | <NA>  474 <NA> |    188   81 <NA> |   -413  393 <NA> |
19  | (*)G.1 | <NA>   26 <NA> |    -33    5 <NA> |     67   21 <NA> |
20  | (*)G.2 | <NA>   24 <NA> |     34    5 <NA> |    -68   19 <NA> |
21  | (*)A.1 | <NA>   41 <NA> |   -108   12 <NA> |   -170   29 <NA> |
22  | (*)A.2 | <NA>   52 <NA> |    -14    0 <NA> |   -172   52 <NA> |
.          .     .    .    .       .    .    .       .    .    .
.          .     .    .    .       .    .    .       .    .    .
```

Las categorías adicionales se han señalado con un `*`, no tienen ni masa (`mass`), ni valores de inercia (`inr`), ni contribuciones a los ejes principales (`ctr`).

Capítulo 19: Análisis de correspondencias conjunto

Para obtener el mapa del ACCo de la imagen 19.3, simplemente tenemos que cambiar la opción `lambda` por `"JCA"`. Dado que los ejes no están anidados, no se dan los porcentajes de inercia de los diferentes ejes, solamente se dan para el resultado global de todo el espacio.

```
summary(mjca(womenG[,1:4], lambda = “JCA”))
```

```
Principal inertias (eigenvalues):
```

```
1         0.353452
2         0.128616
3         0.015652
4         0.003935
          --------
Total:    0.520617
```

```
Diagonal inertia discounted from eigenvalues: 0.125395
Percentage explained by JCA in 2 dimensions: 90.2%
(Eigenvalues are not nested)
[Iterations in JCA: 31 , epsilon = 9.33e-05]

Columns:

      name   mass  qlt  inr      k=1  cor  ctr     k=2  cor ctr
1   |  Q1.1 |  183  969   21 |    204  693   22 |  -129  276  24 |
2   |  Q1.2 |   35  803   23 |    144   61    2 |   503  742  69 |
3   |  Q1.3 |    6  557   32 |    163    9    0 |  1260  548  71 |
4   |  Q1.4 |   26  992  137 |  -1637  991  201 |   -45    1   0 |
5   |  Q2.1 |   13  597   31 |    394  125    6 |  -764  471  60 |
6   |  Q2.2 |   95  956   26 |    250  431   17 |  -276  525  56 |
.       .     .    .    .        .    .    .       .    .   .
.       .     .    .    .        .    .    .       .    .   .
```

En el ACCo, las correlaciones al cuadrado, y en consecuencia también las calidades, son todas mucho mayores.

En el resultado del ACCo la inercia «total» es la inercia de la matriz de Burt modificada, que incluye una parte debida a las matrices modificadas de la diagonal. Para obtener la inercia de las matrices situadas fuera de la diagonal, tenemos que restar de la inercia total la "`Diagonal inertia discounted from eigenvalues: 0.125395`". Dado que la solución buscada es bidimensional y que por construcción ajusta los valores de las matrices de la diagonal, los primeros dos valores propios también contienen esta parte adicional, que tenemos que descontar. La proporción de inercia (de fuera de la diagonal) explicada es, por tanto:

$$\frac{0,3534+0,1286-0,1254}{0,5206-0,1254}=0,9024$$

es decir, el porcentaje del 90,2% indicado anteriormente [véase el apéndice teórico (A.32)]. El valor del denominador de la expresión anterior, el total ajustado 0,5206 – 0,1254 = 0,3952, también lo podemos obtener como:

$$\text{inercia de } \mathbf{B}-\frac{J-Q}{Q}=1,1452-\frac{12}{16}=0,3952$$

Para obtener la solución del ACM ajustado, es decir, las mismas coordenadas estándares del ACM y (casi) los mismos factores de escala («casi» óptimos ya que mantenemos el anidamiento, lo que no ocurre con el ajuste), escribiremos lo siguiente (no es necesario especificar la opción `lambda "adjusted"` ya que es la opción por defecto):

```
summary(mjca(womenG[,1:4]))

Principal inertias (eigenvalues):

dim      value      %     cum%   scree plot
1        0.349456   66.3  66.3   ************************
2        0.123157   23.4  89.7   *********
3        0.023387    4.4  94.1   *
4        0.005859    1.1  95.2

Adjusted total inertia: 0.526963

Columns:

      name   mass   qlt  inr      k=1   cor  ctr     k=2  cor ctr
1   |  Q1.1 |  183   996   22 |    210   687   23 |  -141  309  30 |
2   |  Q1.2 |   35   822   26 |    145    53    2 |   549  769  85 |
3   |  Q1.3 |    6   562   38 |    165     8    0 |  1394  554  91 |
4   |  Q1.4 |   26  1009  141 |  -1680  1008  214 |   -51    1   1 |
5   |  Q2.1 |   13   505   36 |    412   119    6 |  -743  387  59 |
6   |  Q2.2 |   95   947   27 |    253   424   17 |  -281  522  61 |
.      .      .     .    .        .     .    .       .    .   .
.      .      .     .    .        .     .    .       .    .   .
```

Calculamos la inercia total ajustada, utilizada en los porcentajes anteriores, a partir de la expresión (19.5) de la página 200. Y calculamos las dos primeras inercias principales ajustadas (valores propios) a partir de la expresión (19.6) [véanse también (A.35) y (A.36)].

Capítulo 20: Propiedades del escalado óptimo del ACM

En el capítulo 20 generalizamos las ideas que vimos en los capítulos 7 y 8 al caso multivariante. En este capítulo utilizaremos los datos sobre ciencia y medio ambiente que se hallan disponibles en nuestro paquete **ca**. Para cargar estos datos basta con que ejecutemos la instrucción:

```
data(wg93)
```

El *data frame* resultante `wg93` contiene los resultados de las cuatro preguntas descritas en la página 206, así como los de tres variables demográficas: género, edad y educación (las dos últimas con seis categorías cada una de ellas). Después de salvar los resultados del ACM en el objeto `wg93.mca`, podemos obtener el mapa del ACM de la imagen 20.1 de la siguiente manera:

```
wg93.mca <- mjca(wg93[,1:4], lambda = "indicator")
plot(wg93.mca, what = c("none", "all"))
```

El mapa resultante tiene el primer y el segundo eje invertidos, pero —como dijimos anteriormente— esto no tiene consecuencia alguna.

Obtuvimos la tabla de la imagen 20.2 calculando las contribuciones al eje 1 de una matriz de 5 × 4 (primero calculamos las coordenadas principales `wg93.F`, luego calculamos las contribuciones de las filas `wg93.coli`):

```
wg93.F    <- wg93.mca$colcoord %*% diag(sqrt(wg93.mca$sv))
wg93.coli <- diag(wg93.mca$colmass) %*% wg93.F^2
matrix(round(1000*wg93.coli[,1]/wg93.mca$sv[1]^2, 0), nrow = 5)
```

```
     [,1]  [,2]  [,3]  [,4]
[1,]  115   174   203    25
[2,]   28    21     6     3
[3,]   12     7    22     9
[4,]   69    41    80     3
[5,]   55    74    32    22
```

En las siguientes instrucciones calculamos las primeras coordenadas estándares como las puntuaciones de los cuatro ítems de cada uno de los 871 encuestados, así como la puntuación media:

```
Ascal <- wg93.mca$colcoord[1:5,1]
Bscal <- wg93.mca$colcoord[6:10,1]
Cscal <- wg93.mca$colcoord[11:15,1]
Dscal <- wg93.mca$colcoord[16:20,1]
As    <- Ascal[wg93[,1]]
Bs    <- Bscal[wg93[,2]]
Cs    <- Cscal[wg93[,3]]
Ds    <- Dscal[wg93[,4]]
AVEs  <- (AS+Bs+Cs+Ds)/4
```

Situando en una misma matriz las puntuaciones anteriores, podemos calcular sus correlaciones al cuadrado mediante la función `cor()`:

La función de correlación `cor()`

```
cor(cbind(As,Bs,Cs,Ds,AVEs))^2
```

```
              As           Bs          Cs           Ds        AVEs
As    1.000000000  0.139602528  0.12695057  0.005908244  0.5100255
Bs    0.139602528  1.000000000  0.18681032  0.004365286  0.5793057
Cs    0.126950572  0.186810319  1.00000000  0.047979010  0.6273273
Ds    0.005908244  0.004365286  0.04797901  1.000000000  0.1128582
AVEs  0.510025458  0.579305679  0.62732732  0.112858161  1.0000000
```

En la última fila (o columna) aparecen las correlaciones al cuadrado (en análisis de homogeneidad diríamos valores de discriminación) de la página 210. Su media proporciona la inercia principal de la matriz binaria.

```
sum(cor(cbind(As,Bs,Cs,Ds,AVEs))[1:4,5]^2)/4
[1] 0.4573792

wg93.mca$sv[1]^2
[1] 0.4573792
```

La función de covarianza `cov()`

Otro resultado, no mencionado en el capítulo 20, es que el ACM también maximiza la covarianza media entre las puntuaciones de los cuatro ítems. Para verlo, primero, calculamos la matriz de covarianzas de 4×4 de las puntuaciones (dado que la función `cov()` calcula las covarianzas habituales «no sesgadas», dividiendo por $N - 1$, multiplicando por $(N - 1)/N$ obtenemos las covarianzas «sesgadas»). Luego calculamos el valor medio de los 16 valores utilizando la función `mean()`:

```
cov(cbind(As,Bs,Cs,Ds,AVEs)) * 870/871

             As          Bs         Cs          Ds
As   1.11510429  0.44403796  0.4406401  0.04031951
Bs   0.44403796  1.26657648  0.5696722  0.03693604
Cs   0.44064007  0.56967224  1.3715695  0.12742741
Ds   0.04031951  0.03693604  0.1274274  0.24674968

mean(cov(cbind(As,Bs,Cs,Ds)) * 870/871)
[1] 0.4573792
```

Fijémonos en que la suma de las varianzas de las puntuaciones de los cuatro ítems es igual a 4:

```
sum(diag(cov(cbind(As,Bs,Cs,Ds)) * 870/871))
[1] 4
```

En (20.2) calculamos las varianzas individuales y su media sobre toda la muestra:

```
VARs <- ((As-AVEs)^2 + (Bs-AVEs)^2 + (Cs-AVEs)^2 + (Ds-AVEs)^2)/4
mean(VARs)
[1] 0.5426208
```

que es la pérdida de homogeneidad: 1 menos la primera inercia principal.

Suprimiendo las etiquetas de las filas, podemos obtener el mapa de la imagen 20.3 como un mapa `"rowprincipal"` (ejecutando `help(plot.ca)` podemos visualizar las opciones para hacer diagramas):

```
plot(wg93.mca, map = "rowprincipal", labels = c(0,2))
```

Actualmente, el análisis de subgrupos del capítulo 21 solamente se halla en la función `ca()`. Sin embargo, dado que con la función `mjca()` podemos obtener la matriz de Burt, es fácil hacer el ACM de subgrupos con la matriz de Burt. Empecemos con un análisis AC de subgrupos de vocales y consonantes con los datos de los autores contenidos en el paquete **ca**.

```
data(author)
vowels <- c(1,5,9,15,21)
consonants <- c(1:26)[-vowels]
summary(ca(author, subsetcol = consonants))

Principal inertias (eigenvalues):

       dim     value       %     cum%  scree plot
 [1,]  1       0.007607   46.5  46.5  ************************
 [2,]  2       0.003253   19.9  66.3  ***********
 [3,]  3       0.001499    9.2  75.5  *****
 [4,]  4       0.001234    7.5  83.0  ****
  .    .         .           .     .
  .    .         .           .     .
[12,]          --------  -----
[13,]  Total:  0.016371  100.0

Rows:

      name   mass  qlt  inr      k=1  cor  ctr     k=2  cor ctr
1   |  td( |   85   59   29 |     7    8    1 |  -17   50    7 |
2   |  d() |   80  360   37 |   -39  196   16 |  -35  164   31 |
3   |  lw( |   85  641   81 |  -100  637  111 |    8    4    2 |
4   |  ew( |   89  328   61 |    17   27    4 |   58  300   92 |
.      .       .    .    .       .    .    .      .    .   .
.      .       .    .    .       .    .    .      .    .   .

Columns:

      name   mass  qlt  inr      k=1  cor  ctr     k=2  cor ctr
1   |    b |   16  342   21 |   -86  341   15 |   -6    2    0 |
2   |    c |   23  888   69 |  -186  699  104 |  -97  189   66 |
3   |    d |   46  892  101 |   168  783  171 |  -63  110   56 |
4   |    f |   19  558   33 |  -113  467   33 |  -50   91   15 |
.      .       .    .    .       .    .    .      .    .   .
.      .       .    .    .       .    .    .      .    .   .

summary(ca(author, subsetcol = vowels))

Principal inertias (eigenvalues):

      dim     value       %     cum%  scree plot
[1,]  1       0.001450   63.7  63.7  ************************
[2,]  2       0.000422   18.6  82.3  ******
```

```
[3,]  3       3e-04000   13.2   95.5  ****
[4,]  4       0.000103    4.5  100.0
[5,]          --------  -----
[6,]  Total:  0.002276  100.0

Rows:

     name   mass  qlt  inr    k=1  cor  ctr    k=2  cor  ctr
1  |  td( |   85  832  147 |   58  816  195 |    8   15   13 |
2  |  d() |   80  197   44 |  -12  118    9 |  -10   79   20 |
3  |  lw( |   85  235   33 |   14  226   12 |   -3    9    2 |
4  |  ew( |   89  964  109 |   31  337   60 |   42  627  382 |
.     .      .    .    .      .    .    .      .    .    .
.     .      .    .    .      .    .    .      .    .    .

Columns:

     name   mass  qlt  inr    k=1  cor  ctr    k=2  cor  ctr
1  |    a |   80  571   79 |    9   34    4 |  -35  537  238 |
2  |    e |  127  898  269 |   67  895  393 |    4    3    5 |
3  |    i |   70  800  221 |  -59  468  169 |   50  332  410 |
4  |    o |   77  812  251 |  -79  803  329 |   -8    9   12 |
5  |    u |   30  694  179 |  -71  359  105 |  -69  334  335 |
```

Ahora vamos a realizar el ACM de subgrupos versión Burt que vimos en las páginas 220-221 con los datos sobre el trabajo de las mujeres que hemos guardado en `womenG` después de eliminar los valores perdidos de las variables demográficas (págs. 308-309). Primero utilizamos la función `mjca()` para obtener la matriz de Burt, a continuación aplicaremos el AC de subgrupos al cuadrante de la matriz de Burt reacomodada sin datos perdidos (tabla de la imagen 21.3). Hacemos la selección definiendo un vector de índices que llamamos `subset`:

```
womenG.B <- mjca(womenG)$Burt
subset   <- c(1:16)[-c(4,8,12,16)]
summary(ca(womenG.B[1:16,1:16], subsetrow = subset,
     subsetcol = subset))

Principal inertias (eigenvalues):

dim      value      %      cum%  scree plot
1        0.263487   41.4   41.4  ************************
2        0.133342   21.0   62.4  ************
3        0.094414   14.9   77.3  *********
4        0.047403    7.5   84.7  *****
5        0.032144    5.1   89.8  ***
6        0.026895    4.2   94.0  ***
7        0.019504    3.1   97.1  **
8        0.013096    2.1   99.1  *
9        0.005130    0.8   99.9
```

```
10       0.000231     0.0   100.0
11       0.000129     0.0   100.0
         --------   -----
Total:   0.635808   100.0
```

```
Rows:
     name    mass  qlt  inr       k=1  cor  ctr      k=2  cor  ctr
1  | Q1.1 |  183  592   25 |    -228  591   36 |     11    1    0 |
2  | Q1.2 |   35  434   98 |     784  345   81 |   -397   88   41 |
3  | Q1.3 |    6  700  119 |    2002  306   88 |   2273  394  224 |
4  | Q2.1 |   13  535  113 |   -1133  236   65 |   1276  299  162 |
5  | Q2.2 |   95  452   69 |    -442  421   71 |   -119   30   10 |
6  | Q2.3 |  120  693   64 |     482  688  106 |    -40    5    1 |
7  | Q3.1 |   28  706  114 |   -1040  412  114 |    878  294  160 |
8  | Q3.2 |  152  481   38 |    -120   91    8 |   -249  390   71 |
9  | Q3.3 |   47  748  106 |     990  681  175 |    312   67   34 |
10 | Q4.1 |  143  731   49 |    -390  702   83 |     80   29    7 |
11 | Q4.2 |   66  583   84 |     582  414   84 |   -371  168   68 |
12 | Q4.3 |    7  702  119 |    1824  312   90 |   2041  391  222 |
```

La secuencia de instrucciones que, por regresión lineal, permiten modificar la escala para obtener el mejor ajuste de las tablas situadas fuera de la diagonal de la matriz de Burt no es fácil de hacer. Como es bastante larga no la incluimos en este apéndice. Sin embargo, la podemos encontrar en la página web. Esperamos incorporarla próximamente en el paquete **ca**.

Capítulo 22: Análisis de tablas cuadradas

Tal como vimos en el capítulo 21, el AC de matrices asimétricas cuadradas consiste en dividir la tabla en una parte simétrica y en una parte antisimétrica, y luego llevar a cabo el AC en la parte simétrica y un AC sin centrar en la parte antisimétrica, con los mismos pesos y distancias χ^2. En la matriz compuesta mostrada en (22.4) podemos realizar simultáneamente ambos análisis. Después de leer la tabla de movilidad en el *data frame* `mob`, las secuencias de instrucciones para formar la matriz compuesta y luego llevar a cabo el AC son las siguientes. (Antes de llevar a cabo el análisis, tenemos que transformar `mob` en una matriz. En caso contrario no podríamos combinar de forma adecuada filas y columnas para crear la matriz compuesta `mob2`.):

```
mob  <- as.matrix(mob)
mob2 <- rbind(cbind(mob,t(mob)), cbind(t(mob),mob))
summary(ca(mob2))
```

```
Principal inertias (eigenvalues):
dim      value      %      cum%  scree plot
 1       0.388679   24.3   24.3  ************************
 2       0.232042   14.5   38.8  ***************
 3       0.158364    9.9   48.7  **********
```

```
 4       0.158364     9.9    58.6  **********
 5       0.143915     9.0    67.6  *********
 6       0.123757     7.7    75.4  ********
 7       0.081838     5.1    80.5  *****
 8       0.070740     4.4    84.9  *****
 9       0.049838     3.1    88.0  ***
10       0.041841     2.6    90.6  ***
11       0.041841     2.6    93.3  ***
12       0.022867     1.4    94.7  *
13       0.022045     1.4    96.1  *
14       0.012873     0.8    96.9  *
15       0.012873     0.8    97.7  *
16       0.010360     0.6    98.3  *
17       0.007590     0.5    98.8  *
18       0.007590     0.5    99.3  *
19       0.003090     0.2    99.5
20       0.003090     0.2    99.7
21       0.001658     0.1    99.8
22       0.001148     0.1    99.9
23       0.001148     0.1    99.9
24       0.000620     0.0    99.9
25       0.000381     0.0   100.0
26       0.000381     0.0   100.0
27       0.000147     0.0   100.0
         --------   -----
Total:   1.599080   100.0
```

Rows:

```
     name   mass   qlt   inr      k=1   cor   ctr      k=2   cor ctr
1   | Arm |   43   426    54 |   -632   200    44 |    671   226   84 |
2   | Art |   55   886   100 |   1521   793   327 |    520    93   64 |
3   | Tcc |   29    83    10 |   -195    73     3 |     73    10    1 |
4   | Cra |   18   293    32 |    867   262    34 |   -298    31    7 |
.       .     .     .     .        .     .     .        .     .    .
.       .     .     .     .        .     .     .        .     .    .
15  | ARM |   43   426    54 |   -632   200    44 |    671   226   84 |
16  | ART |   55   886   100 |   1521   793   327 |    520    93   64 |
17  | TCC |   29    83    10 |   -195    73     3 |     73    10    1 |
18  | CRA |   18   293    32 |    867   262    34 |   -298    31    7 |
.       .     .     .     .        .     .     .        .     .    .
.       .     .     .     .        .     .     .        .     .    .
```

Columns:

```
     name   mass   qlt   inr      k=1   cor   ctr      k=2   cor ctr
1   | ARM |   43   426    54 |   -632   200    44 |    671   226   84 |
2   | ART |   55   886   100 |   1521   793   327 |    520    93   64 |
3   | TCC |   29    83    10 |   -195    73     3 |     73    10    1 |
4   | CRA |   18   293    32 |    867   262    34 |   -298    31    7 |
.       .     .     .     .        .     .     .        .     .    .
.       .     .     .     .        .     .     .        .     .    .
```

```
15 | Arm |  43  426   54 |  -632  200   44 |   671  226  84 |
16 | Art |  55  886  100 |  1521  793  327 |   520   93  64 |
17 | Tcc |  29   83   10 |  -195   73    3 |    73   10   1 |
18 | Cra |  18  293   32 |   867  262   34 |  -298   31   7 |
.      .     .    .    .       .    .    .       .    .   .
.      .     .    .    .       .    .    .       .    .   .
```

Las inercias principales coinciden con los valores de la tabla de la imagen 22.4. Las dos primeras dimensiones corresponden a la parte simétrica de la matriz. Las dimensiones 3 y 4, con valores propios repetidos, corresponden a la parte antisimétrica. Podemos observar que las coordenadas de las dos primeras dimensiones aparecen repetidas en dos bloques. Por defecto la función `summary()` proporciona sólo las dos primeras dimensiones. Si queremos más dimensiones tenemos que especificarlo. Por ejemplo, para obtener las cuatro primeras dimensiones:

```
summary(ca(mob2, nd = 4))
```

```
Rows:
```

```
    name      k=3  cor ctr      k=4  cor ctr
1   | Arm |   -11    0   0 |   416   87  47 |
2   | Art |    89    3   3 |   423   61  62 |
3   | Tcc |  -331  211  20 |   141   38   4 |
4   | Cra |  -847  250  80 |    92    3   1 |
.      .       .    .   .       .    .   .
.      .       .    .   .       .    .   .
15  | ARM |    11    0   0 |  -416   87  47 |
16  | ART |   -89    3   3 |  -423   61  62 |
17  | TCC |   331  211  20 |  -141   38   4 |
18  | CRA |   847  250  80 |   -92    3   1 |
.      .       .    .   .       .    .   .
.      .       .    .   .       .    .   .
```

```
Columns:
```

```
    name      k=3  cor ctr      k=4  cor ctr
1   | ARM |  -416   87  47 |   -11    0   0 |
2   | ART |  -423   61  62 |    89    3   3 |
3   | TCC |  -141   38   4 |  -331  211  20 |
4   | CRA |   -92    3   1 |  -847  250  80 |
.      .       .    .   .       .    .   .
.      .       .    .   .       .    .   .
15  | Arm |   416   87  47 |    11    0   0 |
16  | Art |   423   61  62 |   -89    3   3 |
17  | Tcc |   141   38   4 |   331  211  20 |
18  | Cra |    92    3   1 |   847  250  80 |
.      .       .    .   .       .    .   .
.      .       .    .   .       .    .   .
```

Para las dimensiones 3 y 4 de las filas observamos que también se repiten las coordenadas en dos bloques, pero en este caso con los signos cambiados. Asimismo podemos observar que los valores de las coordenadas de las columnas de la tercera dimensión son los de las filas de la cuarta dimensión cambiadas de signo, y que los de las columnas de la cuarta dimensión son los de las filas de la tercera dimensión también cambiadas de signo. En cualquier caso para obtener el mapa sólo necesitamos uno de los dos conjuntos de coordenadas. Recordemos, como vimos en el capítulo 22, que estos mapas tienen una forma propia de interpretación.

Capítulo 23: Recodificación de datos

En el capítulo 23 vimos transformaciones simples de datos y la posterior aplicación del AC habitual. Como ilustración del análisis de datos continuos, utilizaremos los indicadores de la Unión Europea. Supongamos que hemos leído los datos y que los hemos introducido en un objeto llamado `EU`. A continuación convertimos los datos en rangos (utilizando la función `rank()` de R, una vez aplicada la útil función `apply()` para obtener `EUr`). Finalmente realizamos el doblado (para obtener `EUd`) de la siguiente manera:

Conversión a rangos mediante la función `rank()`

```
EUr <- apply(EU, 2, rank)-1
EUd <- cbind(EUr, 11-EUr)
colnames(EUd) <- c(paste(colnames(EU), "-", sep=""),
                   paste(colnames(EU), "+", sep=""))
EUd
```

```
   Unemp- GDPH-  PCH-    PCP-   RULC- Unemp+ GDPH+  PCH+    PCP+   RULC+
Be      6     6     6     6.5    4.5      5     5     5     4.5    6.5
De      4    11    10     0.0    7.0      7     0     1    11.0    4.0
Ge      2    10    11     5.0    6.0      9     1     0     6.0    5.0
Gr      5     1     1     1.0   11.0      6    10    10    10.0    0.0
Sp     11     3     3    10.0    2.0      0     8     8     1.0    9.0
Fr      7     8     8     3.5    4.5      4     3     3     7.5    6.5
Ir     10     2     2    11.0    1.0      1     9     9     0.0   10.0
It      9     7     7     9.0    9.0      2     4     4     2.0    2.0
Lu      0     9     9     3.5    8.0     11     2     2     7.5    3.0
Ho      8     5     4     6.5    3.0      3     6     7     4.5    8.0
Po      1     0     0     8.0    0.0     10    11    11     3.0   11.0
UK      3     4     5     2.0   10.0      8     7     6     9.0    1.0
```

Fijémonos en que hemos introducido los nombres de las columnas con la función `paste()`. Finalmente, ejecutando `ca(EUd)` obtenemos el mapa de la imagen 23.5.

Capítulo 24: Análisis de correspondencias canónico

Con el paquete **ca** no podemos obtener los resultados del capítulo 24. Sin embargo, los podemos obtener utilizando el programa XLSTAT (descrito más adelante) o el paquete **vegan** de Jari Oksanen (véanse los recursos web en el apéndice bibliográfico). Este último recurso no solamente desarrolla el ACC, sino que también permite ejecutar el AC y el ACP (pero sin muchas de las opciones que tenemos en el paquete **ca**). Dado que este paquete se utiliza a menudo en el contexto de

datos sobre ecología, como por ejemplo los del capítulo 24, hablaremos de estaciones o localidades (muestras), «especies» y «variables» (explicativas). La utilización de **vegan** es tan fácil como la de **ca**. La principal función es `cca()`, que la podemos utilizar con los siguientes dos formatos:

```
cca(X, Y, Z)
cca(X ~ Y + condition(Z))
```

donde `X` es la matriz de recuentos de localidades × especies, `Y` es la matriz de localidades × variables de datos explicativos y `Z` es la matriz de localidades × variables de datos condicionados por si queremos hacer (de forma opcional) un ACC parcial. El segundo formato es un formato tipo regresión. Nosotros utilizaremos el primer formato. Si sólo especificamos `X`, se lleva a cabo un análisis del AC (lo podemos comprobar con uno de los ejemplos anteriores. Por ejemplo con `summary(cca(author))` para comparar los resultados con los que hemos obtenido anteriormente —los libros serían las «localidades» y las letras las «especies»—. Por defecto, por ejemplo ejecutando `plot(cca(autor))`, obtenemos el mapa que hemos llamado `"colprincipal"`). Si especificamos `X` e `Y`, realizamos un ACC. Si especificamos `X`, `Y` y `Z`, el análisis es un ACC parcial.

Supongamos que el *data frame* `bio` contiene los datos biológicos que vimos en los capítulos 10 y 24 en una tabla de 13 × 92, y que `env` contiene las variables `logBa`, `logFe` y `logPE` en una tabla de 13 × 3. Podemos llevar a cabo el ACC de la manera siguiente:

```
summary(cca(bio, env))

Call:
cca(X = bio, Y = env)

Partitioning of mean squared contingency coefficient:
Total 0.7826
Constrained 0.2798
Unconstrained 0.5028

Eigenvalues, and their contribution to the
             mean squared contingency coefficient

             CCA1    CCA2     CCA3     CA1     CA2      CA3
lambda     0.1895  0.0615  0.02879  0.1909  0.1523  0.04159
accounted  0.2422  0.3208  0.35755  0.2439  0.4385  0.49161

               CA4      CA5      CA6      CA7      CA8      CA9
lambda     0.02784  0.02535  0.02296  0.01654  0.01461  0.01076
accounted  0.52719  0.55957  0.58891  0.61004  0.62871  0.64245
```

```
Scaling 2 for species and site scores
--- Species are scaled proportional to eigenvalues
--- Sites are unscaled: weighted dispersion equal on all dimensions

Species scores

               CCA1      CCA2      CCA3        CA1       CA2       CA3
Myri_ocul  0.1732392  0.245915 -0.070907  0.6359626 -0.063479  0.031990
Chae_seto  0.5747974 -0.270816  0.011814 -0.5029157 -0.674207  0.093354
Amph_falc  0.2953878 -0.114067  0.075979 -0.2224138  0.041797 -0.005020
Myse_bide -0.5271092 -0.505262 -0.103978 -0.0789909  0.176683 -0.484208
Goni_macu -0.1890403  0.122783 -0.044679 -0.1045244  0.030134  0.111827
Amph_fili -0.9989672 -0.075696  0.107184 -0.3506103  0.076968  0.004931
    .       .          .         .         .          .         .
    .       .          .         .         .          .         .

Site constraints (linear combinations of constraining variables)

         CCA1       CCA2       CCA3
S4   -0.06973    0.75885  -2.29951
S8   -0.35758    1.47282   2.27467
S9    0.48483   -0.72459  -0.66547
S12   0.02536    0.27129  -0.14677
S13   0.30041   -0.01531  -0.80821
S14   0.79386    1.16229   0.24314
S15   0.96326   -0.88970   0.14630
S18  -0.16753    0.25048  -0.77451
S19   0.36890   -0.81800   1.50620
S23  -0.09967   -1.90159   0.06877
S24   0.05478    0.96184  -0.10635
R40  -3.71393   -0.20698   0.53031
R42  -2.96641   -0.18264  -0.67736

Biplot scores for constraining variables

         CCA1       CCA2      CCA3
logBa  0.9957   -0.08413  0.03452
logFe  0.6044   -0.72088  0.33658
logPE  0.4654    0.55594  0.68710
```

Fijémonos en lo siguiente:

— el `mean squared contingency coefficient` es la inercia total;

— en el espacio restringido, los encabezamientos de las inercias principales son `CCA1`, `CCA2`, etc., y en el espacio no restringido, son `CA1`, `CA2`, etc.;

— todos los porcentajes se expresan en relación a la inercia total;

— `Scalin 2` significa filas (localidades) en coordenadas estándares, y columnas (especies) en coordenadas principales, es decir, equivale a las coordenadas de `"colprincipal"` de la función `plot.ca()`;

— las `Species scores` son coordenadas principales de columnas;

— los `Site constraints` son las coordenadas estándares de filas;

— los `Biplot scores for constraining variables` son los coeficientes de correlación ponderados entre las variables explicativas y las coordenadas de las localidades.

Capítulo 25: Consideraciones sobre estabilidad e inferencia

En el capítulo 25 llevamos cabo varios automuestreos de tablas con el fin de investigar su variabilidad, así como pruebas de permutaciones para contrastar hipótesis nulas. Por ejemplo, obtuvimos el mapa del AC del automuestreo parcial de los datos sobre los autores que mostramos en la imagen 25.1 y 25.2 de la siguiente manera (hemos insertado comentarios). Si nos fijamos en la imagen 25.1, sólo hemos dibujado 100 de las 1000 réplicas; como el muestreo es aleatorio, los resultados no serán exactamente iguales:

Muestreo aleatorio multinomial utilizando `rmultinom()`

```
data(author)
author.ca <- ca(author)
nsim <- 1000
# cálculo de la suma de las filas
author.rowsum <- apply(author, 1, sum)
# cálculo de las nsim simulaciones del primer libro
author.sim <- rmultinom(nsim, author.rowsum[1], prob = author[1,])
# cálculo de las nsim simulaciones de los otros libros y
  combinación de columnas
for (i in 2:12) {
    author.sim <- cbind(author.sim,
                  rmultinom(nsim, author.rowsum[i],
                  prob = author[i,]))
    }
# transposición para tener el mismo formato que la
  matriz original
author.sim <-  t(author.sim)
author.sim2 <- matrix(rep(0, nsim*12*26), nrow = nsim*12)
# reorganización de filas para juntar las matrices
for (k in 1:nsim) {
    for (i in 1:12) {
         author.sim2[(k-1)*12+i,] <- author.sim[k+(i-1)*nsim,]
         }
    }
```

Utilizando la fórmula de transición, a partir de las coordenadas estándares de filas calculamos las coordenadas principales simuladas de las columnas:

```
# obtención de las coordenadas estándares de las filas
author.rowsc <- author.ca$rowcoord[,1:2]
# cálculo de las coordenadas principales de todas las réplicas
  mediante la fórmula de transición
```

```
author.colsim <- t(t(author.rowsc) %*% author.sim2[1:12,])/
                 apply(author.sim2[1:12,], 2, sum)
for (k in 2:nsim) {
    author.colsim <- rbind(author.colsim, t(t(author.rowsc) %*%
                     author.sim2[((k-1)*12+1):(k*12),])/
                     apply(author.sim2[((k-1)*12+1):(k*12),], 2, sum))
# reorganización de las coordenadas de las filas para que
  todas las letras estén juntas
author.colsim2 <- matrix(rep(0, nsim*26*2), nrow = nsim*26)
for (j in 1:26) {
    for (k in 1:nsim) {
         author.colsim2[(j-1)*nsim+k,] <- author.colsim[j+(k-1)*26,]
         }
    }
```

Representación gráfica de los puntos y de los perímetros convexos:

```
# representación de todos los puntos (para etiquetarlos
  utilizamos el primer formato)
plot(author.colsim[,1], -author.colsim[,2], xlab = "dim1",
     ylab = "dim2", type = "n")
text(author.colsim[,1], -author.colsim[,2], letters, cex = 0.5,
     col = "gray")
# representación de los perímetros convexos de cada letra
# en primer lugar calculamos las coordenadas principales de
  las letras de la matriz original
author.col <- t(t(author.rowsc) %*% author)/
              apply(author, 2, sum)
for (j in 1:26) {
    points <- author.colsim2[(nsim*(j-1)+1):(nsim*j),]
# en todos estos mapas invertimos la segunda coordenada
    points[,2] <- -points[,2]
    hpts <- chull(points)
    hpts <- c(hpts,hpts[1])
    lines(points[hpts,], lty = 3)
    text(author.col[j,1], -author.col[j,2],
         letters[j], font = 2, cex = 1.5)
    }
```

Finalmente llevamos a cabo el recorte de todos los perímetros convexos hasta eliminar el 5% de los puntos de las proyecciones de las nubes, luego representamos los perímetros convexos recortados:

```
plot(author.colsim2[,1], -author.colsim2[,2], xlab = "dim1",
    ylab = "dim2", type = "n")
for (j in 1:26) {
    points <- author.colsim2[(nsim*(j-1)+1):(nsim*j),]
# en todos estos mapas invertimos la segunda coordenada
    points[,2] <- -points[,2]
```

```
    repeat {
          hpts <- chull(points)
          npts <- nrow(points[-hpts,])
          if(npts/nsim < 0.95) break
          points <- points[-hpts,]
          }
    hpts <- c(hpts,hpts[1])
    lines(points[hpts,], lty = 3)
    text(author.col[j,1], -author.col[j,2], letters[j],
          font = 2)
    }
```

Para representar las elipses de confianza tenemos que bajar el paquete **ellipse** de la página web de R (www.R-project.org). El texto del programa para representar las elipses de confianza utilizando las réplicas del automuestreo es el siguiente:

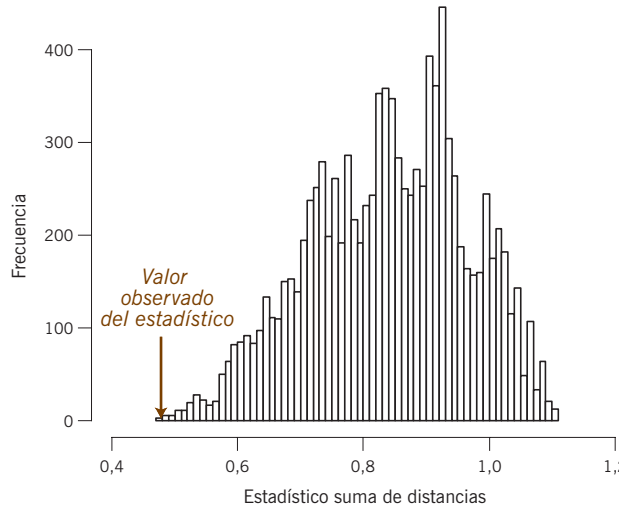
```
# elipses de confianza – necesitamos el paquete 'ellipse'
plot(author.colsim2[,1], -author.colsim2[,2],xlab = "dim1",
    ylab = "dim2", type = "n")
for (j in 1:26) {
    points <- author.colsim2[(nsim*(j-1)+1):(nsim*j),]
    # en todos estos mapas invertimos la segunda coordenada
    points[,2] <- -points[,2]
    covpoints <- cov(points)
    meanpoints <- apply(points, 2, mean)
    lines(ellipse(covpoints, centre = meanpoints))
    text(author.col[j,1], -author.col[j,2], letters[j],
          font = 2)
    }
```

Para reproducir las elipses de confianza del mapa de la imagen 25.3 obtenidas a partir del método Delta, necesitamos la matriz de covarianzas de las coordenadas principales estimadas. La podemos obtener utilizando el programa SPSS. En la página de la red CARME, www.carme-n.org, podemos obtener más detalles y programas adicionales.

Pruebas de permutaciones

Vamos a realizar una prueba de permutación con los datos sobre los autores. Para ello, consultamos un listado de todas las $11 \times 9 \times 7 \times 5 \times 3 = 10395$ combinaciones posibles de los pares libro-autor de los mapas del AC. Luego calcularemos las sumas de las distancias entre los pares del mismo autor en el mapa del AC. En la página web podemos encontrar el programa R para la obtención de todas las combinaciones posibles. El programa hace un listado de los 11 primeros pares posibles {(1,2), (1,3), (1,4), ..., (1,12)}, luego un listado de los 9 pares posibles de cada uno de ellos, por ejemplo para (1,2) son {(3,4), (3,5), ..., (3,12)}, luego un listado de los 7 pares posibles de cada uno de éstos últimos, y así suce-

Imagen B.6:
Distribución exacta, suponiendo cierta la hipótesis nula, del estadístico suma de distancias en la prueba de permutaciones para contrastar la aleatoriedad de las posiciones de los pares de textos del mismo autor en el mapa del AC. El valor observado es el segundo más pequeño de todos los 10395 valores posibles



sivamente. En la imagen B.6 mostramos la distribución de las 10395 distancias posibles en la que hemos señalado la distancia correspondiente a la combinación observada 0,4711. Como vimos en el capítulo 25, no hay otra combinación de pares libro-autor en el mapa del AC bidimensional con una suma de distancias menor. Por tanto, el valor *p* asociado con este valor es igual a 1/10395, es decir, $p < 0{,}0001$. Hicimos una prueba similar en los mapas del AC de subgrupos, que mostramos en las imágenes 21.1 (sólo de consonantes) y 21.2 (sólo de vocales) y obtuvimos 47 y 67 combinaciones a la izquierda del valor observado, por lo que los valores *p* son 48/10395 = 0,0046 y 68/10395 = 0,0065, respectivamente.

Pruebas de permutaciones en ACC

En el ACC nos centramos en la parte del espacio de las variables respuesta (en general en ecología, las especies), que está relacionado linealmente con un determinado conjunto de variables explicativas (en general variables ambientales), como puede verse en el capítulo 24. Pero, ¿cómo saber si las variables respuesta están realmente relacionadas con las variables explicativas? Una medida de la relación existente entre ambos conjuntos de variables es la inercia del espacio restringido. Podríamos situar esta inercia en la distribución de inercias del espacio restringido bajo el supuesto de que no hay relación alguna. Podemos obtener esta distribución permutando al azar los casos (filas) en la matriz de variables explicativas (o variables respuesta). Al permutar aleatoriamente las filas tendrían que perderse las posibles relaciones de éstas con las filas en la matriz de respuestas. Repetimos el ACC y volvemos a calcular la inercia del espacio restringido. Haciendo esto 999 veces (o el número de veces necesarias para poder calcular un valor *p* con suficiente precisión), podemos situar el valor de inercia observa-

do en la distribución para ver si este valor es inusualmente elevado. Si se halla en el 5% de los valores más elevados, consideraremos que la relación entre las variables respuesta y las variables explicativas es estadísticamente significativa. Como antes, podemos estimar el valor de *p* haciendo un recuento del número de valores de la distribución de permutaciones mayores que el valor observado (para ser significativo el valor observado tiene que ser suficientemente elevado). El paquete **vegan** incorpora esta prueba, que podemos obtener aplicando la función `anova()` a `cca()`:

```
anova(cca(bio, env))

Permutation test for cca under reduced model

Model: cca(X = bio, Y = env)
          Df     Chisq         F  N.Perm     Pr(>F)
Model      3    0.2798    1.6696    1300    0.03462  *
Residual   9    0.5028
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En realidad, el estadístico utilizado no es la inercia sino un «seudo» estadístico F, como el del análisis de la varianza (para más detalles podemos consultar la documentación sobre **vegan**). Debemos fijarnos en el valor de *p* que nos va a dar la salida. Así, en este caso el valor del estadístico F para el espacio restringido es significativamente elevado ($p = 0{,}03462$).

Programas R de Fionn Murtagh

En su reciente libro *Correspondence Analysis and Data Coding with Java and* R (véase el apéndice bibliográfico), Fionn Murtagh proporciona muchos programas en R para AC, especialmente para la recodificación de datos. Están disponibles en Internet y se pueden bajar de www.correspondances.info. En concreto, en sus páginas 21 a 26, encontramos el único programa en R que permite hacer la agrupación jerárquica utilizando el método de Ward, con la incorporación de pesos, que es exactamente lo que necesitamos para el capítulo 15. Suponiendo que hayamos sido capaces de bajar el programa del web mencionado, que hemos leído la tabla de datos de la imagen 15.3 y que la hemos guardado en el *data frame* `food`, podremos realizar el análisis de grupos de los perfiles de las filas que mostramos en la figura de la imagen 15.5, utilizando la función `hierclust()` de Murtagh de la siguiente manera:

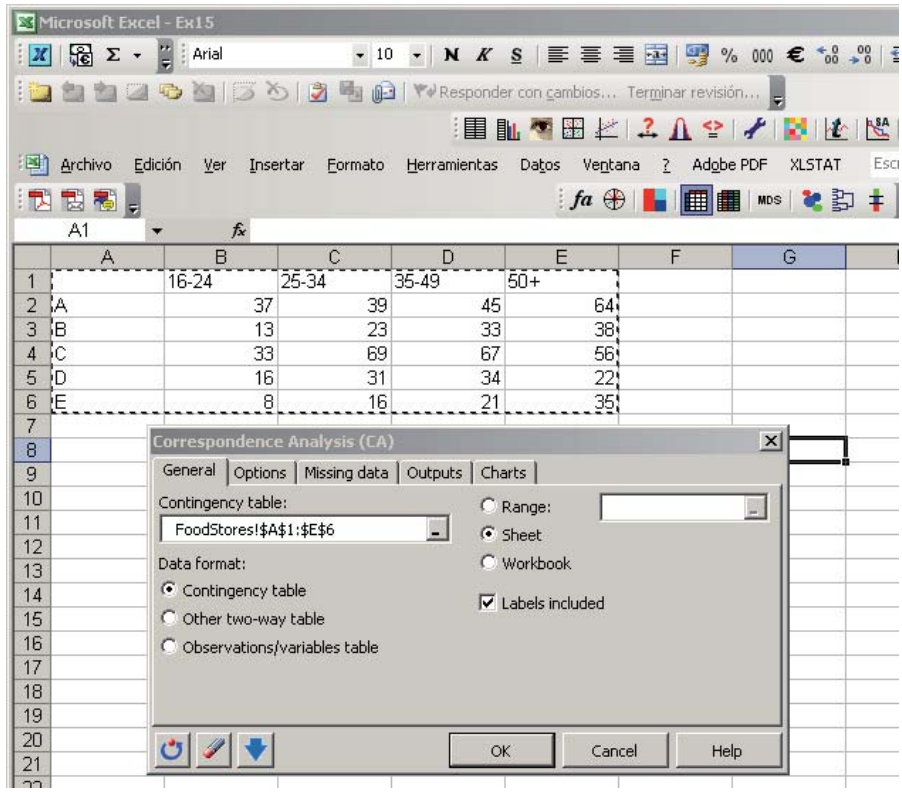
```
food.rpro <- food/apply(food,1,sum)
food.r <- apply(food,1, sum)/sum(food)
food.rclust <- hierclust(food.rpro, food.r)
plot(as.dendrogram(food.rclust))
```

XLSTAT

Para llevar a cabo los análisis de este libro (y análisis adicionales), una de las mejores alternativas es el programa estadístico XLSTAT (www.xlstat.com) derivado de Excel. En XLSTAT, los programas para el AC y el ACM incluyen los ajustes de inercia del ACM y el análisis de subgrupos del AC y del ACM. También incluye un programa para el ACC que incorpora la prueba de la permutación para contrastar que las variables explicativas están significativamente relacionadas con lo ejes principales de la solución restringida. Otros programas para el análisis multivariante de XLSTAT son el análisis de componentes principales, el análisis factorial, el análisis discriminante, el análisis de grupos, la regresión de mínimos cuadrados parcial y el análisis de Procrustes generalizado. Dado que el programa opera en el entorno Excel es muy fácil de utilizar. Por ejemplo, para ejecutar el AC con los datos `food` que hemos utilizado anteriormente en el análisis jerárquico de grupos, clicamos sobre el icono de AC y seleccionamos la tabla (con etiquetas para filas y columnas) que queremos analizar (imagen B.7).

El menú de «Options» permite seleccionar puntos adicionales o subgrupos de datos. El menú «Missing data» permite varias opciones para el tratamiento de los

Imagen B.7:
Menú de XLSTAT para ejecutar el AC en una tabla seleccionada en Excel



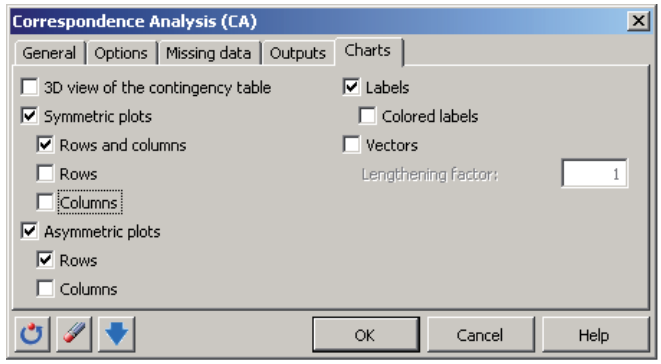


Imagen B.8:
Menú de XLSTAT para seleccionar opciones gráficas del AC

valores perdidos. El menú «Outputs» permite seleccionar varias tablas numéricas (perfiles, distancias χ^2, coordenadas principales, coordenadas estándares, contribuciones, correlaciones al cuadrado, etc.). El menú «Charts» permite llevar a cabo diversos mapas del AC. En la imagen B.8 mostramos cómo llevar a cabo un

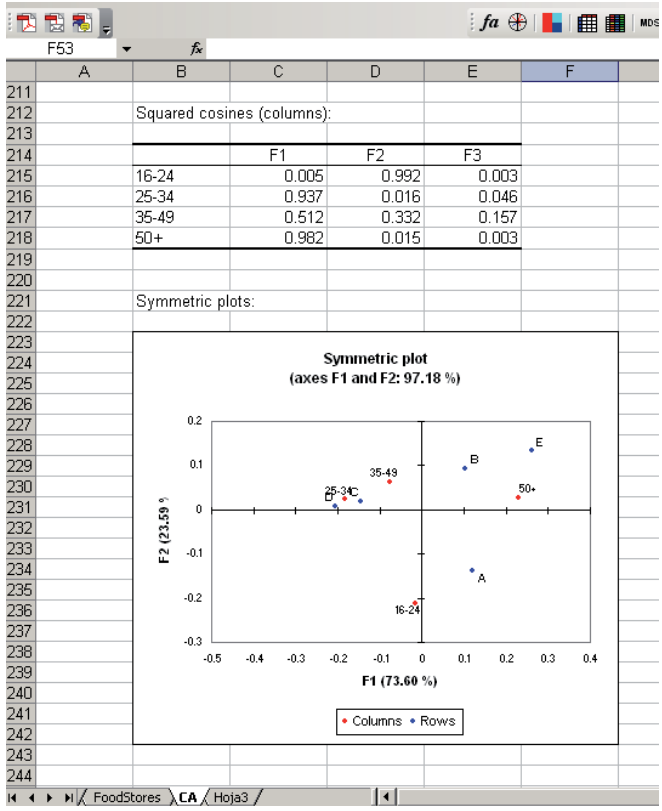


Imagen B.9:
Parte del resultado del programa de AC de XLSTAT, que se proporciona en una hoja de cálculo adicional

mapa simétrico de filas y de columnas, y un mapa asimétrico de filas (es decir, la opción `"rowprincipal"` del paquete **ca**). En la imagen B.9 mostramos parte de estos resultados.

En el módulo de análisis de grupos de XLSTAT podemos desarrollar el análisis de grupos que vimos en el capítulo 15, ya que permite asignar pesos a los puntos; por tanto, podemos realizar la agrupación de Ward de perfiles ponderados con sus masas.

Opciones gráficas

Crear un mapa de AC con determinadas características listo para ser publicado, no es trivial. En esta sección describimos los tres procedimientos utilizados para la obtención de los gráficos de este libro.

Gráficos con LaTeX

La composición tipográfica de la edición en inglés de este libro fue realizada con LaTeX. Con LaTeX —y algunas macros de este programa— podemos crear directamente mapas sin tener que utilizar otros paquetes gráficos. Así, realizamos la mayor parte de los mapas del libro utilizando la macro PicTeX. Como ejemplo de mapa creado en LaTeX, a continuación mostramos el programa que utilizamos para crear el mapa asimétrico correspondiente a los datos de los fumadores que mostramos en la imagen 9.2:

```
\beginpicture
\setcoordinatesystem units <2.5cm,2.5cm>
\setplotarea x from -2.40 to 1.70, y from -1.6 to 2.25
\accountingoff
\gray
\setdashes <5pt,4pt>
\putrule from 0 0 to 1.7 0
\putrule from 0 0 to -1.4 0
\putrule from 0 0 to 0 2.25
\putrule from 0 0 to 0 -1.6
\put {+} at 0 0
\black
\small
\put {Axis 1} [Br] < -.2cm,.15cm> at 1.70 0
\put {0.0748 (87.8\%)} [tr] < -.2cm,-.15cm> at 1.70 0
\put {Axis 2} [Br] < -.1cm,-.4cm> at 0 2.25
\put {0.0100 (11.8\%)} [Bl] <.1cm,-.4cm> at 0 2.25
\setsolid
\putrule from 1.3 -1.3 to 1.4 -1.3
\putrule from 1.3 -1.32 to 1.3 -1.28
\putrule from 1.4 -1.32 to 1.4 -1.28
\put {\it scale} [b] <0cm,.25cm> at 1.35 -1.3
\put {0.1} [t] <0cm,-.2cm> at 1.35 -1.3
\multiput {$\bullet$} at
```

```
 0.06577 0.19373
-0.25896 0.24330
 0.38059 0.01066
-0.23295 -0.05775
 0.20109 -0.07891
 /
\sf
\put {SM} [l] <.15cm,0cm> at   0.06577  0.19373
\put {JM} [r] < -.15cm,0cm> at -0.25896  0.24330
\put {SE} [bl] <.15cm,0cm> at  0.38059  0.01066
\put {JE} [r] < -.15cm,0cm> at -0.23295 -0.05775
\put {SC} [tl] <.15cm,0cm> at  0.20109 -0.07891
\gray
\multiput {$\circ$} at
 1.4384  0.3046
-0.3638 -1.4094
-0.7180 -0.0735
-1.0745  1.9760
 /
\sl
\put {none} [b] <0cm,.2cm> at     1.4384  0.3046
\put {light} [b] <0cm,.2cm> at   -0.3638 -1.4094
\put {medium} [T] <0cm,-.3cm> at -0.7180 -0.0735
\put {heavy} [b] <0cm,.2cm> at   -1.0745  1.9760
\black
\endpicture
```

El programa anterior permite darnos cuenta de que crear un mapa como el de la imagen 9.2 es bastante laborioso. Hay que situar con mucha precisión cada uno de los puntos y de las líneas del mapa. Una ventaja es que podemos asegurar que la razón de escalas del mapa es exactamente 1. Por ejemplo, en este mapa hemos establecido que las unidades en los ejes de coordenadas vertical y horizontal sean exactamente iguales (`2,5 cm`).

Gráficos en Excel

Muchos de los mapas nuevos de esta segunda edición los hemos hecho en Excel, a partir de los resultados del análisis estadístico en XLSTAT. Sin embargo, para asegurar que la razón de escalas de los mapas de los capítulos 17 a 19 fuera correcta, tuvimos que efectuar algunos ajustes. Así tuvimos que redefinir los valores máximos y mínimos de los ejes y alargar vertical u horizontalmente los mapas. Luego los copiamos en metaarchivos y los pegamos en el *Adobe Ilustrator*. Al realizar esta operación se modifica algo la razón de escalas del mapa, las unidades verticales aumentan algo más que las horizontales. Por tanto, de nuevo, tuvimos que retocar los mapas. Posteriormente los guardamos en formato *PostScript Encapsulado* (EPS), para finalmente incorporarlos al texto como archivos LaTeX utilizando la instrucción `\includegraphics`, por ejemplo:

```
\begin{figure}[h]
\center{\includegraphics[width=10cm,keepaspectratio]{Ex18_5.eps}}
\caption{\sl MCA map of Burt matrix of four questions on women
         working, showing first and second dimensions;
         total inertia = 1.145, percentage inertia in map: 65.0\%.}
\end{figure}
```

Gráficos en R

En este libro también hemos creado muchos mapas en R. Por ejemplo, los del capítulo 25. Para incorporarlos al texto, primero los copiamos como metaarchivos, luego los pegamos en el *Adobe Ilustrator*, los terminamos de ajustar para asegurar que la razón de escalas fuera correcta y finalmente los guardamos en formato EPS.