eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifiziala Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Métodos basados en distancias estadísticas en comparación y clasificación de poblaciones

Itziar Irigoien Garbizu

Donostia, 2008

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifiziala Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Métodos basados en distancias estadísticas en comparación y clasificación de poblaciones

Itziar Irigoien Garbizu

Donostia, 2008

Memoria realizada bajo la dirección de

Concepción Arenas Sola

Departamento de Estadística de la Facultad de Biología

Universidad de Barcelona

para optar al grado de Doctora.

Agradecimientos

Quiero empezar dándote las gracias a ti, Conchita, pues sin tu esfuerzo y empeño este trabajo no hubiera sido posible. Además, gracias a tu calidad humana hemos sido capaces de sobrellevar el trabajo cuando mi situación personal no ha sido la más idónea. Gracias a ti Francesc, porque además de tu ayuda con la genética siempre has hecho de «mensajero» cuando ha hecho falta.

Eskerrak eman nahi dizkizuet zuei ere, Arantza eta Inma, izan ere zuek baizarete bide honetan jarri ninduzuenak eta gero ere edozein beharren aurrean beti laguntzeko prest egon zaretenak. Iraide, Leire, zuei ere eskerrik asko, egoera ilundu eta zalantzak sortzen zitzaikidanean hor zinetelako animoak emateko. Eta ez ditut ahaztu nahi egungo lankideak ere: Yosu, Ana, Bittori, Joseba, Itziar, Basi, ..., fakultatean egin didazuen harrera ona eta eguneroko lanean suposatzen duzuen patsadazko irlatxoagatik. Ezin baditut ere denen izenak jarri, denei eskerrik asko.

Asier, Andone, Eyu, Elena, nire eskerrik beroenak zuei esker far egitea ez zaidalako ahazten eta lanean jarraitzeko gogoa ematen didazuelako.

Ezin dut amaitu nire familia aipatu gabe. Olatz eta Bittor, zenbat eta zenbat aldiz zaindu dituzue haurrak, besteak beste lan hau aurrera ateratzeko? Aita eta ama, haurrak zaindu, idatziak gainbegiratu, ingelesa «entsaiatu»,... zer esan..., beti hor zaudete eta hau ez denean bestean laguntzeko. Eta azkenik, Joseba, eskerrik asko lanetik etxera bueltatzea nire bizi-poza bihurtu duzulako.

Aita eta amari

Índice

Capítulo 1

Introducción

1.1. Objetivos planteados

La revolución genética que nace a finales del siglo XX se fundamenta en dos pilares: la secuenciación automática del DNA y la informática. De la fusión de ambas se acuñó el término Bioinformática, que hace referencia de forma general a la aplicación de la informática al análisis de los datos experimentales obtenidos a partir del estudio de seres vivos. Solamente cuando se tuvo capacidad de computación suficiente se pudieron analizar las ingentes cantidades de secuencias nucleotídicas acumuladas. La comparación de dichas secuencias en los diferentes tipos celulares o especies, la estructura de dichas secuencias en los genomas o la expresión de los genes en diferentes tejidos requieren, además de capacidad de cálculo, la implementación de procesos estadísticos clásicos y también el desarro-

llo de nuevos modelos adecuados a los diferentes análisis genéticos.

Desde un punto de vista histórico, a finales de los años cincuenta y principios de los sesenta del siglo pasado, se empezaron a tener secuencias aminoacídicas de proteínas tales como la insulina, la ribonucleasa o el citocromo *c*, entre otras. Los procedimientos de secuenciación de proteínas eran muy complejos y laboriosos (Sanger and Tuppy, 1961; Sanger and Thompson, 1963). Este enfoque experimental se abandonó al hacerse posible la obtención de secuencias nucleotídicas del material genético, el DNA. Con los procedimientos de secuenciación por el método químico de Maxam and Gilbert (1977) y enzimático de Sanger *et al.* (1977), se empezaron a acumular secuencias nucleotídicas de diversos genes. Sin embargo se puede considerar que la bioinformática, tal y como la conocemos actualmente, se inició con la puesta a punto de los métodos semiautomáticos de secuenciación. El hecho de que en 1988 se instituyese la Organización Genoma Humano (HUGO) potenció aún más las tecnologías experimentales. El desarrollo de dicho proyecto trajo como consecuencia la creación de grandes bases de datos como GeneBank, EMBL o DNA Database of Japan, capaces de almacenar las secuencias de DNA para su posterior comparación. Paralelamente surgió la necesidad de desarrollar programas informáticos que permitiesen agregar con facilidad nuevos datos, suministrar las secuencias almacenadas de forma ágil e incluso modificar las secuencias nucleotídicas en caso de tener más información sobre ellas. Con el proyecto Genoma Humano se pretendía tener más información sobre los genes implicados directa o indirectamente en diferentes enfermedades y patologías humanas. Actualmente, se puede considerar que, gracias al proyecto

de secuenciación completa, el ser humano ha pasado a ser un organismo modelo (como la bacteria *Escherichia coli*, el maíz (*Zea mays*) o la mosca de la fruta (*Drosophila melanogaster*)), en cuanto a que ha permitido conocer aspectos generales de la estructura genómica. Las primeras aproximaciones a la secuencia completa de nuestro genoma aparecieron en el año 2001 (Venter *et al.*, 2001; Consortium, 2001) y el anuncio de la obtención definitiva data del 2004 (Consortium, 2004). Toda esta información no hubiese podido ser procesada correctamente sin la ayuda de programas informáticos adecuados ni de ordenadores de elevadas prestaciones. Desde el punto de vista experimental surgió otra técnica analítica que es extremadamente útil para conocer la expresión génica. Esta técnica es la de los microarrays (Kulesh *et al.*, 1987). Con ella puede conocerse la transcripción de millares de genes en diferentes tejidos, ya sean sanos o patológicos (por ejemplo, células cancerosas). También es útil en el estudio del control genético del desarrollo embrionario.

Como ya se ha apuntado, toda esta cantidad ingente de datos necesita de una serie de programas informáticos para ser analizados y obtener una interpretación biológica adecuada. Dichos programas han sido obtenidos a partir de procedimientos estadísticos clásicos así como a partir del desarrollo de nuevas aplicaciones estadísticas. Un problema biomédico frecuente es el poder llevar a cabo la clasificación adecuada de poblaciones. En este contexto biomédico se entiende por población un grupo de individuos con una serie de características comunes (genéticas, sintomatológicas, etc.). Por este motivo la comparación y clasificación de poblaciones utilizando datos multivariantes es un problema crucial en los pro-

cesos biológicos. Uno de los métodos estadísticos más interesantes sobre todo si se trabaja con grandes cantidades de datos, es el análisis cluster. Este procedimiento permite la identificación de grupos de observaciones con características similares y diferenciadas de las de otros grupos. Estos métodos son aplicados en áreas tan diferentes como la documentación, la búsqueda de información en Internet, la identificación de grupos de genes que actúen juntos en un mismo proceso biológico o incluso en los análisis de imágenes, entre muchos otros. En los estudios genéticos, es evidente la necesidad de buscar una clasificación real y precisa de los genes en un determinado número de grupos. Dicha clasificación será esencial para deducir genes implicados en la misma función o bien relacionados con una misma enfermedad. Este conocimiento ha de permitir un diagnóstico más preciso de una determinada patología y concretar más su tratamiento. Son muchos los métodos cluster desarrollados y no es el objetivo de esta memoria el hacer una revisión de los mismos. Sin embargo hay que mencionar la importancia de los métodos jerárquicos tanto acumulativos como divisivos (Eisen *et al.*, 1998; Wen *et al.*, 1998) o los métodos de recolocación interactivos, siendo tal vez el método k-means el más utilizado (Hartigan and Wong, 1979; Herwig *et al.*, 1999; Tavazoie *et al.*, 1999). Algunos métodos cluster suponen asociado un modelo estadístico, generalmente basado en la distribución normal multivariante (McLachlan and Basford, 1988; Fraley and Raftery, 2002). Sin embargo esta suposición puede dar lugar a alguna limitación en su uso, puesto que:

a) Por una parte, no todos los datos seguirán dicha distribución; de hecho, en la mayoría de situaciones biomédicas vamos a encontrarnos con datos de dife-

rente naturaleza. Estos pueden ser continuos (como la variación del color en los microarrays), o ser medidas de atributos binarios (como en los estudios de anomalías genéticas donde el valor 1 indicará el fenotipo normal y el valor 0 el fenotipo patológico), o ser atributos de múltiples estados (como los grupos sanguíneos) o frecuencias (como en el caso de la variabilidad de los *loci* microsatélites del genoma humano al comparar poblaciones). Por ello parece necesario pensar en posibles métodos de cluster que puedan ser utilizados con cualquier tipo de datos.

b) Por otra parte, otro problema que aparece si se supone normalidad multivariante, es el de la dimensionalidad. Si ésta es muy elevada con respecto al número de observaciones, en muchas ocasiones la matriz de covarianzas será singular, provocando que los algoritmos basados en la estimación de máxima verosimilitud fallen.

Todo esto llevó a plantearnos si los métodos clásicos de análisis cluster serían adecuados para tan variado tipo de datos o si sería conveniente pensar en alguno nuevo que permitiera su utilización sin estas restricciones. Con esta idea nos planteamos un primer objetivo: *desarrollar métodos de cluster que fueran válidos para cualquier tipo de datos.*

Un problema antiguo y difícil relacionado con el análisis cluster es la determinación del número «real» de clusters. Mientras que en los métodos jerárquicos se deja normalmente este problema al observador, el cual debe interpretar los árboles de clasificación e identificar en qué puntos se separan los clusters relevantes, en

los problemas de optimización, el número de clusters se fija de antemano y se considera como un parámetro externo y fijo en el algoritmo a utilizar. En la literatura se encuentran numerosos métodos que abordan esta cuestión, pudiéndose encontrar en Dudoit and Fridlyand (2002) una buena y exhaustiva revisión de estos métodos. Cabe destacar, entre otros, los trabajos de Calinski and Harabasz (1974), Fowlkes and Mallows (1983) y Hartigan (1985). Además en Milligan and Cooper (1985) se puede consultar una buena revisión de 30 métodos previamente publicados. Otro trabajo muy interesante es el de Rousseeuw (1987), ya que en él se presenta un método adecuado para cualquier tipo de datos. También son de señalar los trabajos de Krzanowski and Lai (1988) que estudian el problema mediante las variabilidades dentro y entre los clusters; Jain and Dubes (1988) que presentan una modificación del denominado estadístico de Hubert; Tibshirani *et al.* (2001) que proponen el estadístico Gap capaz de detectar situaciones de un único cluster y los más recientes de Biau *et al.* (2007) donde se propone un estimador basado en grafos; Tan *et al.* (2007) que presentan un algoritmo de cluster que incorpora la estimación del número de clusters y Yan and Ye (2007) que utilizan un estadístico que se basa en el estadístico Gap. Sin embargo, entre todas las reglas publicadas, de ninguna se puede afirmar que sea mejor que las otras, ya que algunas de ellas trabajan mejor que las otras únicamente en ciertos ejemplos simulados. Además, todas ellas requieren datos continuos, salvo el método silhouette (Rousseeuw, 1987). A su vez, este método presenta una restricción, ya que no es capaz de detectar la no presencia de estructura de clusters en los datos, presuponiendo siempre la existencia como mínimo de dos clusters. Así que nos pareció interesante plantearnos un segundo objetivo: *la construcción de una*

regla que permitiera determinar el número de clusters trabajando con cualquier tipo de variables y que además fuera capaz de detectar la no presencia de clusters.

Una vez resuelta la cuestión de determinar el número de clusters presentes en una población, aparece una nueva pregunta. Dado un nuevo individuo, se debe decidir si pertenece a alguno de los clusters previamente identificados o si por el contrario es un individuo atípico, en el sentido de que pertenece a algún grupo nuevo y desconocido. El análisis discriminante introducido en 1936 por R.A. Fisher (Fisher, 1936) es tal vez el método más utilizado para clasificar nuevos individuos, dados unos grupos previamente determinados. Sin embargo este método no contempla la posibilidad de que el nuevo individuo sea atípico. Así pues, dado un nuevo individuo debemos prestar especial atención al hecho de decidir si es atípico o si por el contrario pertenece a uno de los clusters previamente identificados y entonces sí que tiene sentido usar una regla de clasificación para decidir a qué cluster pertenece. En la literatura se encuentra solución al problema de la tipicalidad únicamente en situaciones muy concretas. Así, Rao (1962) desarrolla un test de tipicalidad para k poblaciones siguiendo una distribución normal multivariante con matriz de varianzas covarianzas común y parámetros conocidos. En McDonald *et al.* (1976) se trata el caso de muestras pequeñas y datos continuos y en McLachlan (1982) el caso de la mixtura de grupos. En Cuadras and Fortiana (2000) se da solución al problema de la tipicalidad en el caso de dos poblaciones y datos mixtos. Finalmente, Bar-Hen (2001) presenta un test de tipicalidad para k poblaciones siguiendo una distribución normal multivariante con matriz de varianzas-covarianzas común y parámetros conocidos centrando la

atención del trabajo en la estimación de proporciones. Así que nos planteamos un tercer objetivo en relación con esta problemática: *construir un test de tipicalidad válido para cualquier número de grupos y cualquier tipo de variables.*

Otra cuestión que captó nuestra atención fue el tema de las *réplicas* cuyo uso conlleva diferentes dificultades. Hasta hace poco, en los estudios genéticos utilizando microarrays normalmente no se realizaban réplicas del experimento, y cuando sí se realizaban, la práctica común era obtener la media de las réplicas y trabajar con esta media. Ha sido en los últimos años cuando el tema de las réplicas ha empezado a interesar a los investigadores al darse cuenta que replicar no equivale a duplicar los resultados. Es cierto que en condiciones ideales cabe esperar que no existan diferencias entre réplicas o que las diferencias que puedan presentarse se deban al azar. Sin embargo, se ha observado que un estudio más detallado de las réplicas permite valorar la robustez del método experimental (saber si la técnica es muy sensible a mínimos cambios ambientales) y a su vez asegurar que el comportamiento detectado sobre los genes (por ejemplo, si se expresan bajo ciertas condiciones) es un comportamiento constante o si los genes son sensibles a pequeñas variaciones. Por ello ha empezado a interesar el determinar si existen realmente diferencias entre réplicas e intentar extraer la máxima información de las mismas. Una primera aproximación a esta problemática, nos llevó a centrarnos en una situación experimental real y muy determinada. En concreto nos fijamos en situaciones en las que los datos se pueden guardar en matrices $n \times p$ donde las n filas representan los genes sobre los que se hacían p mediciones, suponiendo que estas medidas han sido replicadas, y por tanto para

cada réplica se dispone de una de estas matrices de datos. Como parecía que esta situación con réplicas y matrices de datos no tenia porqué ser exclusiva de un estudio genético donde normalmente las medidas de expresión de los genes son datos continuos, nos propusimos abordar esta situación, pero desde un punto de vista algo más general. Concretamente nuestro siguiente objetivo fue: *dada una situación experimental replicada, donde los datos puedan recogerse en matrices $n \times p$ (una por réplica) y con datos no necesariamente continuos, obtener algún método que permitiera la obtención de clusters para las filas de dichas matrices (equivalentemente para las columnas en caso de ser éstas las que se quisiesen agrupar).*

Dentro de esta misma temática, se planteó un nuevo reto, relativo a un problema que no estaba bien resuelto en la literatura. En los estudios denominados «time course experiments» se mide el nivel de expresión de un conjunto de genes a lo largo de un periodo, en general, muy corto de tiempo. Así en función del perfil del gen, es decir, en función de la forma de la gráfica que se obtiene al poner en el eje de abcisas los instantes de tiempo y en el de ordenadas el nivel de expresión, se pretende agrupar los genes, ya que genes con el mismo perfil representan genes con el mismo patrón de expresión génica. Los algoritmos existentes sobre este tema presentan bastantes limitaciones. Algunos de ellos requieren que el experimentador fije de antemano posibles perfiles a determinar (Zhao *et al.*, 2001; Möller-Levet *et al.*, 2003; Peddada *et al.*, 2003; Luo *et al.*, 2004; Ernst *et al.*, 2005) y además no incluyen el estudio individual de las réplicas utilizando la media de las mismas. Otros autores realizan diferentes propuestas: modelos gau-

sianos teniendo en cuenta las estructuras de correlación entre clusters (Fraley and Raftery, 2002); modelos autoregresivos (Ramoni *et al.*, 2002) o, por ejemplo, la utilización de métodos bayesianos (Heard *et al.*, 2005). Pero parece ser que debido a la complejidad de computación que requieren estas soluciones, su aplicación se está viendo muy limitada y además no son adecuadas para el caso de experimentos replicados. Otras aproximaciones como la presentada por Bar-Joseph *et al.* (2003); Ma *et al.* (2006), utilizan los conceptos de spline, pero nuevamente no son aplicables si los tiempos de observación son muy cortos. Por otra parte, existen muchos métodos que utilizan el coeficiente de correlación o alguna variante del mismo como medida de asociación entre genes (Chu *et al.*, 1998; Heyer *et al.*, 1999). Sin embargo el uso del coeficiente de correlación requiere la utilización de perfiles fijados de antemano y además no siempre es una medida adecuada de asociación entre genes (Peddada *et al.*, 2003). Así que se nos planteó un nuevo objetivo: *desarrollar algún método de cluster para una situación experimental aplicable a «time course experiments» realizados en periodos muy cortos de tiempo, con réplicas, que no requiriera la determinación de perfiles prefijados de antemano y que no presentara ninguna de las contradicciones o limitaciones como las que presentan los métodos basados en el coeficiente de correlación.*

Finalmente, nuestro último objetivo fue *desarrollar un soporte informático en el que estuvieran implementados los procedimientos metodológicos que se plantean en esta memoria.* Para ello, se eligió el software de libre distribución R (`http://www.r-project.org/`) ya que éste ofrece la posibilidad de crear nuevos paquetes y hacerlos accesibles a todos los posibles usuarios. En este caso se

ha creado el paquete que hemos denominado `DBmethods` haciendo referencia al común denominador de esta memoria «Distance Based Methods».

1.2. Organización de la memoria

En esta memoria se abordan todas las cuestiones planteadas anteriormente y se presentan soluciones a las mismas, estando organizada de la forma siguiente:

El capítulo 2 contiene la notación y los resultados previos necesarios para desarrollar los diferentes aspectos de este trabajo. No es por tanto original y va acompañado de las correspondientes citas bibliográficas. Concretamente está dedicado al denominado enfoque basado en distancias y al análisis procrustes.

El capítulo 3 presenta cuatro métodos de cluster basados en el concepto de variabilidad geométrica que se han diseñado en esta memoria. Dichos métodos son aplicables ante cualquier tipo de datos, siendo alguno de ellos una generalización de métodos clásicos aptos únicamente para datos continuos.

El capítulo 4 muestra una nueva metodología para determinar tanto el número «real» de clusters como para detectar observaciones atípicas, sin importar el número de grupos o el tipo de variables.

El capítulo 5 presenta un cluster divisivo sin ningún tipo de restricción sobre el número de objetos a clasificar y una extensión del mismo al caso de tener experimentos replicados. En este capítulo se tratan aquellos experimentos cuyos

datos pueden ser recopilados en matrices de dimensiones $n \times p$, una por cada réplica. Además no se supone ninguna restricción sobre el tipo de datos, es decir, no tienen porqué ser necesariamente continuos.

El sexto y último capítulo, resuelve el tratamiento de las réplicas en los denominados «time course experiments» realizados a lo largo de un periodo de tiempo corto y en los que interesa agrupar genes que se expresan de forma análoga.

A lo largo de la presente memoria todos los resultados previamente conocidos van acompañados de su correspondiente cita bibliográfica, y todos los capítulos con resultados originales (capítulos 3-6) presentan una breve introducción del problema a resolver, el desarrollo de la metodología propuesta, su aplicación a datos reales y simulados, un apartado de conclusiones y también información sobre la difusión de la metodología ante la comunidad científica (presentación en congresos y/o publicaciones). Finalmente, cada capítulo incluye una breve descripción de las funciones implementadas en el paquete `DBmethods` para la aplicación de la metodología presentada y, más tarde, en el anexo A, se ha incluído el manual de todo el paquete.

Además la memoria incluye 2 anexos. El mencionado anexo A con el manual del paquete `DBmethods` donde se describen de forma detallada todas las funciones que se ofrecen en el paquete. Hay que hacer notar que estas funciones requieren de algunas funciones de cálculo que se han denominado auxiliares. Estas funciones auxiliares no se detallan en el anexo A pero sí se detallan en el segundo de los

anexos, el anexo B, su código fuente. El anexo A es presisamente el manual al que el usuario podrá acceder desde el mismo software R cuando cargue el paquete `DBmethods` y el anexo B contiene todos los códigos fuente que se han programado para la creación del paquete.

Capítulo 2

Preliminares: Enfoque basado en distancias y análisis procrustes

En este capítulo se introducen las notaciones y conceptos necesarios para el posterior desarrollo de la memoria. Concretamente está dedicado al denominado enfoque basado en distancias y al análisis procrustes. Todo lo que aparece en este capítulo no es original de la memoria y va acompañado de las correspondientes citas bibliográficas.

2.1. Enfoque basado en distancias

El análisis de datos y el análisis multivariante comparten un mismo objetivo: dado un conjunto de objetos o casos, obtener información de los mismos a partir del análisis de las observaciones realizadas sobre estos objetos. Esta información puede obtenerse básicamente desde el enfoque *objeto*×*variable* o desde el enfoque *basado en distancias*. En el primer caso, los métodos estadísticos utilizan como información la denominada matriz de datos, la cual contiene los valores de las variables observadas (columnas) sobre los objetos (filas). En el enfoque basado en distancias, son precisamente las distancias entre objetos el punto de partida y los métodos estadísticos utilizan únicamente esta información. Es evidente que estos dos enfoques son complementarios, siendo en general sencillo pasar del enfoque objeto×variable al enfoque basado en distancias a través de las distancias estadísticas. El enfoque basado en distancias presenta ciertas ventajas. Algunos de los métodos que analizan la matriz de datos requieren que estos sean continuos, limitación que queda solventada con el uso de métodos basados únicamente en la matriz de distancias, y que por tanto tienen únicamente en consideración las semejanzas o diferencias entre objetos. Por otra parte, el problema de los datos faltantes o missing también queda resuelto de forma satisfactoria con el uso de distancias adecuadas, no siendo en tal caso necesario eliminar objetos con algún valor faltante ni sustituir estos valores faltantes por estimaciones. Además, hay que resaltar que en ciertas circunstancias es más natural trabajar con distancias entre objetos que suponer que los datos siguen alguna distribución de probabilidad determinada, con parámetros muchas veces desconocidos. Podríamos pues

resumir la gran utilidad de los métodos basados en distancias señalando que permiten su utilización con cualquier tipo de datos, sin ninguna restricción sobre la existencia de una posible distribución de probabilidad subyacente.

El enfoque objeto×variable generalmente parte de un conjunto de datos $\mathbf{Z}$ $(n \times p)$ donde cada fila corresponde a un objeto y las columnas a las variables medidas sobre ellos, variables que pueden ser cuantitativas, cualitativas, binarias o mixtas. Por tanto, un conjunto C de n objetos viene representado por n observaciones de una p-variable aleatoria $\mathbf{Z} = (Z_1, \ldots, Z_p)$ con valores en un espacio métrico $\mathcal{R} \subset \mathbf{R}^p$ y función de densidad f respecto a una medida adecuada λ. Representaremos como $\mathbf{z}_i$ cada objeto, $i = 1, \ldots, n$.

A continuación y antes de pasar a comentar en qué consiste el enfoque basado en distancias pasamos a recordar los conceptos básicos de similaridad, disimilaridad y distancia. La similaridad s_{ij} entre dos objetos i, j se define como una función de los valores observados sobre los mismos, de forma que sea simétrica $(s_{ij} = s_{ji})$, y que indique el grado de semejanza entre los objetos. La mayoría de los coeficientes de similaridad son positivos y se ajustan para que tomen valores menores a uno. Sin embargo, algunos como el coeficiente de correlación pueden tomar valores negativos. A cada medida de similaridad se le asocia una medida que toma valores entre 0 y 1, definida por ejemplo, como $\delta_{ij} = \sqrt{1 - s_{ij}}$. Esta medida simétrica recibe el nombre de disimilaridad. Algunos coeficientes de disimilaridad verifican la denominada propiedad métrica $\delta_{ij} \leq \delta_{ik} + \delta_{jk}$ $\forall i, j, k$, en cuyo caso reciben el nombre de distancias.

Existe gran número de coeficientes de disimilaridad y de distancias, algunas más adecuadas que otras para determinado tipo de variables o situación experimental. Una extensiva lista de las mismas puede consultarse en Gower (1985). No es el objetivo de este capítulo recordar todas las posibles distancias, sin embargo, a continuación comentaremos las distancias que aparecerán a lo largo de la memoria.

Sean $\mathbf{z}_i$ y $\mathbf{z}_j$ dos observaciones del p-vector aleatorio $\mathbf{Z}$.

- **Distancia euclídea**

  Se define la distancia euclídea entre los objetos $\mathbf{z}_i$ y $\mathbf{z}_j$ como:

$$\delta_{ij} = \big((\mathbf{z}_i - \mathbf{z}_j)'(\mathbf{z}_i - \mathbf{z}_j)\big)^{1/2}.$$

- **Distancia de Mahalanobis**

  Sea Σ la matriz de varianzas-covarianzas de $\mathbf{Z}$. Se define la distancia de Mahalanobis entre los objetos $\mathbf{z}_i$ y $\mathbf{z}_j$ como:

$$\delta_{ij} = \big((\mathbf{z}_i - \mathbf{z}_j)'\Sigma^{-1}(\mathbf{z}_i - \mathbf{z}_j)\big)^{1/2}.$$

- **Distancia de Bhattacharyya** (Bhattacharyya, 1946)

  Se define la distancia de Bhattacharyya entre los objetos $\mathbf{z}_i$ y $\mathbf{z}_j$ como:

$$\delta_{ij} = \arccos \sum_{l=1}^{p} \sqrt{z_{il} z_{jl}}\,. \tag{2.1}$$

Esta distancia es en realidad una medida de divergencia entre dos distribuciones multinomiales. Es decir, sobre dos poblaciones i y j se considera definida una variable cualitativa con p estados excluyentes que aparecen con probabilidades $\mathbf{z}_i$ y $\mathbf{z}_j$ respectivamente, siendo la distancia de Bhattacharyya la que mide la discrepancia entre ambas poblaciones.

- **Distancia correlación**

  Se define la distancia correlación entre los objetos $\mathbf{z}_i$ y $\mathbf{z}_j$ como:

$$\delta_{ij} = \sqrt{1 - r_{ij}}\,, \tag{2.2}$$

  donde r_{ij} es el coeficiente de correlación de Pearson para $\mathbf{z}_i$ y $\mathbf{z}_j$.

- **Distancia de Gower** (Gower, 1971a)

  La distancia de Gower se utiliza en el caso de variables mixtas. En esta situación cada objeto está caracterizado por p_1 variables continuas, p_2 variables binarias y p_3 variables cualitativas. Se define el coeficiente de similaridad de Gower entre los objetos i y j de la siguiente manera:

$$s_{ij} = \frac{\sum_{l=1}^{p} s_{ijl}}{\sum_{l=1}^{p} m_{ijl}} \tag{2.3}$$

  donde $p = p_1 + p_2 + p_3$; $s_{ijl} = 1 - \frac{|z_{il} - z_{jl}|}{R_l}$ siendo R_l el rango de la l-ésima variable continua; $s_{ijl} = 1$ en el caso de coincidencia del tipo presencia-presencia para la l-ésima variable binaria; $s_{ijl} = 1$ en el caso de coincidencia para la l-ésima variable cualitativa, y finalmente m_{ijl} siempre toma el valor 1

salvo en las p_2 variables binarias en que toma el valor 0 ante una coincidencia del tipo ausencia-ausencia.

A partir de este coeficiente de similaridad (2.3) se define la distancia de Gower entre los objetos i y j como $\delta_{ij} = [2(1 - s_{ij})]^{1/2}$. Hay que comentar que ésta no es la única transformación posible del coeficiente de similaridad que ofrece una distancia entre objetos. Sin embargo, en este trabajo hemos utilizado precisamente ésta porque garantiza que la distancia obtenida es euclídea (Gower, 1966).

- **Distancia de Gower con datos faltantes** (Gower, 1971a)

  La distancia de Gower admite una variante aplicable cuando los datos presentan valores faltantes, que tiene buenas propiedades y funciona de forma correcta (Montanari and Mignari, 1994). Esta nueva distancia se obtiene asignando a cada uno de los coeficientes s_{ijl} un peso w_{ijl} que toma el valor 1 si se pueden comparar los valores correspondientes a la variable l, sobre los objetos i y j. En caso contrario tomará el valor 0. Así la nueva distancia viene definida por

$$\delta_{ij} = [2(1 - \sum_{l} w_{ijl} s_{ijl} / \sum_{k} m_{ijl} w_{ijl})]^{1/2}. \tag{2.4}$$

  Cabe destacar que al haber valores ausentes la distancia δ obtenida no tiene porqué ser euclídea (Montanari and Mignari, 1994).

A continuación pasamos a comentar los fundamentos del enfoque basado en distancias. Sea C un conjunto de n objetos representado por el p-vector aleatorio

$\mathbf{Z}$ de densidad f respecto de una medida adecuada λ y soporte $\mathcal{R}$. Supongamos que δ es una distancia entre los objetos $\mathbf{z}_i$, $i = 1, \ldots, n$, y sea $\mathbf{X}$ $(n \times q)$ la matriz que contiene las coordenadas principales obtenidas de la matriz de distancias (Gower, 1966; Krzanowski and Marriott, 1994). Es decir, las columnas de $\mathbf{X}$ vienen dadas por los vectores propios de $(\mathbf{I} - \mathbf{1}\mathbf{1}'/n)\mathbf{B}(\mathbf{I} - \mathbf{1}\mathbf{1}'/n)$, donde $\mathbf{1} = (1, \ldots, 1)'$, I es la matriz identidad y $\mathbf{B}$ es la matriz cuyos elementos son los valores $-\frac{1}{2}\delta_{ij}^2$ calculados entre todos los pares de objetos. Estos vectores propios se han ordenado según sus correspondientes valores propios en orden decreciente y se han normalizado de manera que la suma de los cuadrados de sus valores es su correspondiente valor propio. Se puede considerar que la matriz $\mathbf{X}$ contiene las coordenadas en dimensión q de n objetos. Estas coordenadas están centradas en el origen y la distancia euclídea entre los objetos i y j coincide con δ_{ij}. Por tanto, si la distancia es euclídea (no existen valores propios negativos) el espacio métrico $(\mathcal{R}, \delta)$ puede ser relacionado con $(\mathbf{R}^q, \delta_{euclidea})$. Es decir, existe una función $\psi : \mathcal{R} \longrightarrow \mathbf{R}^q$, de manera que $\delta_{ij}^2 = \|\psi(\mathbf{z}_i) - \psi(\mathbf{z}_j)\|^2$. Habiendo convertido la matriz inicial de datos $\mathbf{Z}$ en un conjunto de coordenadas con valores reales $\mathbf{X}$, es posible aplicar a esta matriz $\mathbf{X}$ cualquier técnica estándar de análisis multivariante aplicable a datos continuos. Este enfoque, llamado *basado en distancias*, fue introducido por primera vez por Cuadras and Arenas (1990) en el ámbito de la regresión lineal con variables mixtas. Además ha sido utilizado por otros autores, así Cuadras (1992) estudia el problema de análisis discriminante desde este nuevo punto de vista; Cuadras *et al.* (1996) extienden las ideas originales al caso de la regresión no lineal; Legendre and Anderson (1999) aplican este enfoque al análisis de la varianza multifactorial; Anderson and Robinson (2003) y Anderson

and Willis (2003) lo aplican en el contexto del análisis canónico y más recientemente Krzanowski (2004) estudia el problema del análisis multivariante de la varianza desde este enfoque. En Arenas and Cuadras (2002) puede consultarse una revisión de algunos de estos métodos.

Como esta forma de abordar el análisis estadístico de los datos requiere de distancias euclídeas, cuando la distancia con la que se trabaja no lo es, un modo sencillo de obtener distancias euclídeas δ^*_{ij} consiste en transformar las distancias δ_{ij} como sigue: $\delta^*_{ij} = (\delta^2_{ij} + h)^{1/2}\ \forall i, j$, donde h es el valor absoluto del menor de los valores propios de la matriz $\mathbf{B}$ doblemente centrada (Lingoes, 1971; Gower and Legendre, 1986).

En el contexto anterior, cabe preguntarse si las nociones de esperanza, varianza, distancia de un individuo a un grupo, etc. pueden transportarse al espacio métrico $(\mathcal{R}, \delta)$ inicial. Esta idea lleva a las siguientes definiciones:

Definición 2.1. *(Cuadras and Fortiana, 1995)*
Sea C el conjunto de n objetos representado por el p-vector aleatorio $\mathbf{Z}$ de densidad f respecto de una medida adecuada λ y soporte $\mathcal{R}$. Supongamos que δ es una distancia entre los objetos. Se define la variabilidad geométrica *de C respecto δ como:*

$$V_\delta(C) = \frac{1}{2}\int_{\mathcal{R}\times\mathcal{R}} \delta^2(\mathbf{z}_i, \mathbf{z}_j) f(\mathbf{z}_i) f(\mathbf{z}_j) \lambda(d\mathbf{z}_i) \lambda(d\mathbf{z}_j).$$

Obsérvese que esta definición es una variante del coeficiente de diversidad de Rao (Rao, 1982). Cuando δ es la distancia euclídea, y para $p = 1$, la variabilidad geométrica coincide con la varianza, *i.e.*, $V_\delta(C) = VAR(\mathbf{Z})$. Para valores de $p > 1$,

coincide con la variabilidad total de $\mathbf{Z}$, *i.e.*, $V_\delta(C) = tr(VAR(\mathbf{Z}))$. Aun cuando δ no sea euclídea, la variabilidad geométrica es una medida generalizada de la variabilidad de $\mathbf{Z}$ o del conjunto C.

Definición 2.2. *(Cuadras* et al.*, 1997)*

Sean C_1 y C_2 dos conjuntos de n_1 y n_2 objetos y representados, respectivamente, por los p-vectores aleatorios $\mathbf{Z}_1$ de densidad f_1 y $\mathbf{Z}_2$ de densidad f_2 respecto de una medida adecuada λ y soporte $\mathcal{R}$. Supongamos que δ es una distancia entre los objetos. Se define la distancia (al cuadrado) entre C_1 y C_2 *como:*

$$\Delta^2(C_1, C_2) = \int_{\mathcal{R}\times\mathcal{R}} \delta^2(\mathbf{z}_1, \mathbf{z}_2) f_1(\mathbf{z}_1) f_2(\mathbf{z}_2) \lambda(d\mathbf{z}_1) \lambda(d\mathbf{z}_2) - V_\delta(C_1) - V_\delta(C_2).$$

Nótese que esta definición coincide con la diferencia de Jensen tomando como función de divergencia $4V_\delta(C_i)$, $i = 1, 2$ (Rao, 1982).

Definición 2.3. *(Cuadras* et al.*, 1997)*

Sea C el conjunto de n objetos representado por el p-vector aleatorio $\mathbf{Z}$ de densidad f respecto de una medida adecuada λ y soporte $\mathcal{R}$. Consideremos el objeto representado por $\mathbf{z}_0 \in \mathbf{R}^p$. Supongamos que δ es una distancia entre los objetos. Se define la proximidad (al cuadrado) *de $\mathbf{z}_0$ al conjunto C respecto de δ como:*

$$\phi^2_\delta(\mathbf{z}_0, C) = \int_{\mathcal{R}} \delta^2(\mathbf{z}_0, \mathbf{z}) f(\mathbf{z}) \lambda(d\mathbf{z}) - V_\delta(C).$$

A continuación se presentan (sin demostración) algunas propiedades interesantes de estos conceptos que utilizaremos en posteriores capítulos.

Propiedad 2.1. *(Cuadras* et al.*, 1997)*

En las mismas condiciones que las supuestas en las definiciones anteriores, supongamos que $E(\psi(\mathbf{Z}_i))$ *y que* $E(\|\psi(\mathbf{Z}_i)\|^2)$ *(*$i = 1, 2$*) son finitas, entonces se verifican las siguientes igualdades:*

$$V_\delta(C_i) = E(\|\psi(\mathbf{Z}_i) - E(\psi(\mathbf{Z}_i))\|^2) \quad (i = 1, 2)\,, \tag{2.5}$$

$$\Delta^2(C_1, C_2) = \|E(\psi(\mathbf{Z}_1)) - E(\psi(\mathbf{Z}_2))\|^2, \tag{2.6}$$

$$\phi^2_\delta(\mathbf{z}_0, C_i) = \|\psi(\mathbf{z}_0) - E(\psi(\mathbf{Z}_i))\|^2 \quad (i = 1, 2)\,. \tag{2.7}$$

La función de proximidad da pie a definir en $(\mathcal{R}, \delta)$ el concepto análogo al concepto de media definido en $(\mathbf{R}^q, \delta_{euclidea})$. Denominaremos δ-media al valor $\mathbf{z}_0$ del espacio inicial tal que su imagen por la función ψ coincida con $E(\psi(\mathbf{Z}))$, es decir, $\psi(\mathbf{z}_0) = E(\psi(\mathbf{Z}))$. Por simplificar, se identifican los valores de $\mathbf{z}_0$ con el de $E(\psi(\mathbf{Z}))$.

Estos resultados nos garantizan que las definiciones que acabamos de exponer recuperan las ideas de esperanza, varianza, distancia entre poblaciones y distancia de un punto a una población usuales.

A la hora de aplicar los métodos multivariantes, la función de distancia δ suele considerarse como un dato pero sin embargo las distribuciones de probabilidad de las poblaciones suelen ser desconocidas. Esto lleva a tener que considerar estimadores de los conceptos anteriores y que se obtienen a partir de la distancia δ. Sean $\mathbf{z}_{(1)1}, \ldots, \mathbf{z}_{(1)n_1}$ y $\mathbf{z}_{(2)1}, \ldots, \mathbf{z}_{(2)n_2}$ muestras de dos conjuntos de datos C_1 y C_2, respectivamente. Se construyen de manera natural los siguientes estimadores:

- Variabilidad geométrica

$$\hat{V}_\delta(C_r) = \frac{1}{2n_r^2} \sum_{i,j \in C_r} \delta^2(\mathbf{z}_{(r)i}, \mathbf{z}_{(r)j}), \quad r = 1, 2 \tag{2.8}$$

- Distancia (al cuadrado) entre C_1 y C_2

$$\hat{\Delta}^2(C_1, C_2) = \frac{1}{n_1 n_2} \sum_{\substack{i \in C_1 \\ j \in C_2}} \delta^2(\mathbf{z}_{(1)i}, \mathbf{z}_{(2)j}) - \hat{V}_\delta(C_1) - \hat{V}_\delta(C_2) \tag{2.9}$$

- Proximidad (al cuadrado) de $\mathbf{z}_0$ a C_r, $r = 1, 2$

$$\hat{\phi}_\delta^2(\mathbf{z}_0, C_r) = \frac{1}{n_r} \sum_{i \in C_r} \delta^2(\mathbf{z}_0, \mathbf{z}_{(r)i}) - \hat{V}_\delta(C_r), \quad r = 1, 2 \tag{2.10}$$

Obsérvese que el cálculo de estos estimadores requiere únicamente conocer los valores de las distancias δ_{ij}. Además las propiedades (2.5), (2.6) y (2.7) en versión muestral vienen expresadas por (Cuadras *et al.*, 1997):

$$\begin{aligned}
\hat{V}_\delta(C_r) &= \frac{1}{n_r} \sum_{i=1}^{n} \|\psi(\mathbf{z}_{(r)i})\|^2 - \|\overline{\psi(\mathbf{z}_{(r)})}\|^2 \\
&= \frac{1}{n_r} \sum_{i=1}^{n} \|\mathbf{x}_{(r)i}\|^2 - \|\bar{\mathbf{x}}_{(r)}\|^2, \quad r = 1, 2, \qquad (2.11) \\
\hat{\Delta}^2(C_1, C_2) &= \|\overline{\psi(\mathbf{z}_{(1)})} - \overline{\psi(\mathbf{z}_{(2)})}\|^2 \\
&= \|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(2)}\|^2, \qquad (2.12) \\
\hat{\phi}_\delta^2(\mathbf{z}_0, C_r) &= \|\psi(\mathbf{z}_0) - \overline{\psi(\mathbf{z}_{(r)})}\|^2 \\
&= \|\mathbf{x}_0 - \bar{\mathbf{x}}_{(r)}\|^2, \quad r = 1, 2, \qquad (2.13)
\end{aligned}$$

donde $\psi : \mathcal{R} \longrightarrow \mathbf{R}^q$ representa la función que proporciona las coordenadas principales; $\mathbf{x}_{(r)i} = \psi(\mathbf{z}_{(r)i})$; $\bar{\mathbf{x}}_{(r)}$ es la media de todos los valores de $\psi(\mathbf{z}_{(r)i})$, es decir, es el centro de C_r, con $r = 1, 2$ y finalmente $\mathbf{x}_0 = \psi(\mathbf{z}_0)$.

A fin de simplificar la notación y hacer la lectura de la memoria más ágil, cuando no haya confusión de cuál es la distancia δ que se está utilizando, suprimiremos δ de los subíndices de $V_\delta(C)$ y $\phi^2_\delta(\cdot\,, C)$. Además, en las estimaciones prescindiremos del símbolo circunflejo.

2.2. Análisis procrustes

Fueron Hurley and Catell (1962) quienes denominaron análisis procrustes a la técnica introducida por Mosier (1939). En un principio esta técnica se desarrolló para su utilización en al análisis factorial, empezando a tomar relevancia como técnica multivariante con los trabajos de Schönemann and Carroll (1970) y Gower (1971b). Cabe destacar como principales referencias sobre este tema las debidas a Mosier (1939), Green (1952), Hurley and Catell (1962), Cliff (1966), Schönemann (1966, 1968), Schönemann and Carroll (1970), Kristof and Wingersky (1971), Gower (1975), ten Berge (1977), Sibson (1978), Goodall (1991) o más recientemente Gower and Dijksterhuis (2004) y ten Berge (2006) entre otros. Los métodos procrustes consisten básicamente en ajustar dos o más matrices de datos mediante transformaciones de una de ellas, a fin de que sean lo más parecidas posibles. Su nombre proviene de una leyenda griega en la que se narra la siguiente historia (Humbert, 1988):

> *Procusto tenía estatura y fuerza prodigiosas y atraía a su mansión a los viandantes para robarles y hacerles sufrir suplicios atroces. Les tendía sobre un lecho de hierro y si sus piernas excedían los límites*

del mismo, cortaba de un hachazo la porción sobrante; si, por el contrario, las piernas resultaban más cortas las estiraba hasta que dieran la longitud del lecho fatal.

Realmente la idea básica del análisis procrustes es la de la leyenda: dadas dos o más configuraciones de puntos en un espacio p-dimensional que normalmente representarán medidas tomadas sobre un mismo conjunto de objetos, se trata de averiguar cuán diferentes son estas configuraciones. Esta pregunta tiene sentido después de aproximar estas configuraciones lo máximo posible por traslación, rotación/reflexión y dilatación. El proceso de construir una nueva configuración de esta forma es el análisis procrustes y la suma de cuadrados residual que se obtiene después de aproximar las configuraciones es el denominado estadístico procrustes. De forma resumida el análisis procrustes se presenta a continuación.

2.2.1. El análisis procrustes

Sean $\mathbf{X}_1$ y $\mathbf{X}_2$ dos matrices de datos o configuraciones que se quieren comparar, donde cada fila de las dos matrices se refiere al mismo objeto.

Para poder comparar las dos configuraciones se necesitan ciertas trasformaciones preliminares que hagan posible la comparación. En ese sentido supondremos que estas configuraciones ya están centradas y estandarizadas de forma que son conmensurables entre ellas ($tr(\mathbf{X}_1'\mathbf{X}_1) = tr(\mathbf{X}_2'\mathbf{X}_2)$) (Gower and Dijksterhuis, 2004).

Existen diferentes formulaciones para comparar dos configuraciones pero nosotros nos centraremos en la siguiente. Se buscarán la rotación $\mathbf{T}$ y dilatación por un factor s de la configuración $\mathbf{X}_1$ que mejor ajusten dicha configuración a la configuración $\mathbf{X}_2$. Por tanto, se formula el problema como sigue:

Encontrar la rotación $\mathbf{T}$ y el factor de dilatación s que minimizan

$$\|s\mathbf{X}_1\mathbf{T} - \mathbf{X}_2\|, \tag{2.14}$$

donde $\|s\mathbf{X}_1\mathbf{T} - \mathbf{X}_2\| = tr(s\mathbf{X}_1\mathbf{T} - \mathbf{X}_2)'(s\mathbf{X}_1\mathbf{T} - \mathbf{X}_2)$ [1].

La solución a este problema se obtiene utilizando la descomposición en valores singulares de $\mathbf{X}_2'\mathbf{X}_1$. Sea $\mathbf{X}_2'\mathbf{X_1} = \mathbf{U\Sigma V'}$ tal descomposición. Entonces, la solución al problema viene dada por:

$$\mathbf{T} = \mathbf{VU}' \quad \text{y} \quad s = tr(\mathbf{X}_2'\mathbf{X_1T})/\|\mathbf{X}_1\|.$$

Definición 2.4.

El valor mínimo de (2.14) es el denominado estadístico procrustes *y se denota por* M.

El estadístico procrustes toma valores entre 0 y 1. Valores cercanos a 0 indican una fuerte asociación entre las configuraciones mientras que valores cercanos a 1 indican ausencia de asociación. Sibson (1978) probó que (2.14) define una métrica de manera que estos valores pueden ser tratados como distancias (al cuadrado) entre dos configuraciones.

[1] En lo que sigue, utilizaremos $\|\mathbf{A}\|$ para referirnos a la norma $tr(\mathbf{A}'\mathbf{A})$.

2.2.2. El análisis procrustes generalizado

Sean $\mathbf{X}_1, \ldots, \mathbf{X}_R$, R matrices de datos o configuraciones que se quieren comparar, de manera que cada fila de las matrices se refiere al mismo objeto. Otra vez, supongamos que las transformaciones preliminares ya están realizadas y por tanto que ya están centradas y estandarizadas de forma que son conmensurables entre ellas ($tr(\mathbf{X}_r'\mathbf{X}_r) = tr(\mathbf{X}_u'\mathbf{X}_u), \quad r, u = 1, \ldots, R$).

Para medir el grado de asociación entre las R configuraciones existen varias formulaciones. En esta memoria se ha considerado el enfoque que permite las operaciones de rotación y dilatación de las configuraciones. En este sentido, se formula el problema como sigue:

Encontrar las rotaciones $\mathbf{T}_r$ y factores de dilatación s_r $(r = 1, \ldots, R)$ que minimizan

$$\sum_{r<u} \|s_r\mathbf{X}_r\mathbf{T}_r - s_u\mathbf{X}_u\mathbf{T}_u\|. \tag{2.15}$$

Antes de pasar a recordar la solución de este problema queremos señalar que (2.15) admite la siguiente igualdad:

$$\sum_{r<u} \|s_r\mathbf{X}_r\mathbf{T}_r - s_u\mathbf{X}_u\mathbf{T}_u\| = R\sum_{r=1}^{R} \|s_r\mathbf{X}_r\mathbf{T}_r - \mathbf{X}\mathbf{M}_R\|, \tag{2.16}$$

con $\mathbf{X}\mathbf{M}_R = \frac{1}{R}\sum_{r=1}^{R}(s_r\mathbf{X}_r\mathbf{T}_r)$. De esta manera, la discrepancia entre las R configuraciones se puede entender como la suma de las discrepancias de cada configuración $\mathbf{X}_r$ a una configuración media $\mathbf{X}\mathbf{M}_R$. Ahora es fácil probar la si-

guiente identidad fundamental

$$\sum_{r=1}^{R} \|s_r \mathbf{X}_r \mathbf{T}_r\| = \sum_{r=1}^{R} \|s_r \mathbf{X}_r \mathbf{T}_r - \mathbf{X}\mathbf{M}_R\| - R\|\mathbf{X}\mathbf{M}_R\|.$$

Esta identidad se puede interpretar como una partición del total de la suma de cuadrados (Total = Residual + Medio) y por tanto, minimizar (2.15) se puede considerar como una minimización de la suma de cuadrados residual.

Minimizar (2.15) tiene la solución degenerada $s_r = 0, \quad r = 1, \ldots, R$ y por ello debe imponerse alguna restricción para evitarla. La restricción más frecuente es la de igualar el tamaño de cada configuración antes y después de la transformación. Es decir,

$$\sum \|s_r \mathbf{X}_r \mathbf{T}_r\| = \sum \|\mathbf{X}_r\|.$$

Supuesto que las matrices de rotaciones $\mathbf{T}_r$ $(r = 1, \ldots, R)$ estuvieran determinadas, el mínimo (2.15) se alcanza seleccionando el vector de escalas $\mathbf{s} = (s_1, \ldots, s_R)'$ como el vector propio asociado al mayor de los valores propios de la matriz

$$\mathbf{S} = \left(trace(\mathbf{T}_r \mathbf{X}_r' \mathbf{X}_u \mathbf{T}_u)\right)_{r,u=1,\ldots,R}.$$

Por otra parte, el mínimo de (2.15), o equivalentemente de (2.16), se alcanza cuando se alcanza el mínimo de $\|s_r \mathbf{X}_r \mathbf{T}_r - \mathbf{X}\mathbf{M}_R\|$. Éste es, al fin y al cabo, un problema de procrustes para el caso de $R = 2$ pero donde la configuración $\mathbf{X}\mathbf{M}_R$ es desconocida. Todo esto lleva a que en general se utilizan algoritmos iterativos para hallar las rotaciones $\mathbf{T}_r$ y factores de escala s_r $(r = 1, \ldots, R)$. En Gower and Dijksterhuis (2004) se muestran diferentes algoritmos de manera detallada y bien referenciada.

Definición 2.5.

El valor mínimo de (2.15) se denomina estadístico procrustes generalizado *y se denota por* M_R.

El estadístico procrustes generalizado mide la similaridad que tienen las configuraciones, de manera que cuando las configuraciones son iguales, salvo traslación, dilatación y rotación, el valor de M_R es 0.

Capítulo 3

Métodos de cluster basados en el concepto de variabilidad geométrica

3.1. Introducción

Como ya se ha comentado en el capítulo anterior, el enfoque basado en distancias ha sido aplicado a distintos campos del análisis multivariante, pero no hay ninguna referencia sobre su aplicación al análisis cluster. Si bien hay múltiples algoritmos de cluster que parten directamente de las interdistancias entre los objetos (pueden consultarse por ejemplo en Kaufman and Rousseeuw (1990) o en Gordon (1999)), éstos no establecen las relaciones que tienen con la variabilidad

del conjunto o el clásico enfoque objeto×variable. Además Gower and Krzanowski (1999) señalan la conexión entre las sumas de distancias euclídeas y la variabilidad en un conjunto de objetos, obteniendo una partición de la variabilidad análoga a la identidad fundamental del Análisis de la Varianza (Variabilidad Total = Variabilidad *Dentro* + Variabilidad *Entre*) pero no aplican estos resultados en el contexto del cluster. Por todo ello el objetivo de este capítulo es abordar el tema del análisis de clusters desde el enfoque basado en distancias y en particular desarrollar diferentes métodos utilizando el concepto de variabilidad geométrica. Los métodos que se desarrollan en este capítulo recibirán el nombre de *métodos GEVA* (métodos basados en la GEometric VAriability).

Concretamente, en este capítulo se desarrollan cuatro algoritmos de cluster. Un algoritmo divisivo para la clasificación de los objetos en k clusters, dado un $k > 1$ específico y tres algoritmos aglomerativos. Se comentan sus propiedades y además se incluye un ejemplo real donde se aplican estos algoritmos y se comparan los resultados obtenidos. Se finaliza con una descripción de las funciones implementadas en el paquete `DBmethods` a fin de poder aplicar todo lo presentado en este capítulo de la memoria.

3.2. Algoritmos GEVA

A continuación pasamos a desarrollar los métodos cluster basados en el concepto de variabilidad geométrica.

3.2.1. GEVA - Algoritmo politético divisivo

Sea C un conjunto formado por n objetos, que se quieren clasificar en k clusters C_r de tamaños n_r respectivamente ($r = 1, \ldots, k$), para un valor $k > 1$ fijado con anterioridad. Se cumple la siguiente descomposición de la variabilidad del conjunto C.

Propiedad 3.1. *(Gower and Krzanowski, 1999)*
La variabilidad geométrica del conjunto C dada en la expresión (2.8) se descompone de manera similar a la identidad fundamental del Análisis de la Varianza,

$$\begin{aligned} nV_\delta(C) &= \sum_{r=1}^{k} n_r V_\delta(C_r) + \frac{\mathbf{n}'\Delta\mathbf{n}}{n} \\ \textit{Var. TOTAL} &= \textit{Var. DENTRO} + \textit{Var. ENTRE} \end{aligned} \tag{3.1}$$

donde $\mathbf{n} = (n_1, \ldots, n_k)'$ y Δ es una matriz simétrica $(k \times k)$ con $\Delta_{rs} = \frac{1}{2}\Delta^2(C_r, C_s)$, $r, s = 1, \ldots, k$. Es decir, cada elemento de la matriz es la mitad de la distancia (al cuadrado) definida en (2.9) y calculados entre todos los pares de clusters $C_1, \ldots, C_k$.

A partir de esta descomposición de la variabilidad geométrica del conjunto C establecemos el siguiente criterio para obtener la partición de C en los clusters C_r, $r = 1, \ldots, k$.

Criterio:

Dado un conjunto C de n objetos y fijado un valor $k > 1$, se toma como

partición aquella que minimiza la varibilidad geométrica dentro de los clusters

$$\sum_{r=1}^{k} n_r V_\delta(C_r), \tag{3.2}$$

o equivalentemente, aquella que maximiza la variabilidad entre los clusters,

$$\frac{\mathbf{n}'\Delta\mathbf{n}}{n}. \tag{3.3}$$

Volviendo de nuevo al enfoque basado en distancias, es fácil comprobar que se cumple la siguiente propiedad:

Propiedad 3.2.

Sea C un conjunto de n objetos representado por un p-vector aleatorio $\mathbf{Z}$ de densidad f respecto de una medida adecuada λ y soporte $\mathcal{R}$, sobre el que está definida una distancia δ entre los objetos $\mathbf{z}_i$, $i = 1, \ldots, n$. Sea $\mathbf{X}$ la correspondiente matriz $(n \times q)$ de coordenadas principales y denotemos por $\mathbf{X}_r$ la matriz $n_r \times q$ formada por las r filas de $\mathbf{X}$ asociadas a los objetos clasificados en C_r, $r = 1, \ldots, k$. La descomposición (3.1) de la variabilidad geométrica de $\mathbf{X}$ viene dada por:

$$nV_\delta(C) = \sum_{r=1}^{k} n_r V_\delta(C_r) + \frac{1}{n} \sum_{1 \le r < s \le k} n_r n_s \|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_{(s)}\|^2,$$

donde $\bar{\mathbf{x}}_{(r)}$ es el centro del cluster C_r, $r = 1, \ldots, k$.

Demostración:

Aplicando el resultado (2.12) a todos los pares de clusters C_r y C_s $(r, s = 1, \ldots, k)$ se tiene que los elementos de la matriz Δ de la expresión (3.1) son,

$$\Delta_{rs} = \frac{1}{2}\|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_{(s)}\|^2, \quad r, s = 1, \ldots, k.$$

Sustituyendo este resultado en (3.1) y desarrollando el segundo de los sumandos se obtiene:

$$\begin{aligned}
\frac{1}{n}\mathbf{n}'\Delta\mathbf{n} &= \frac{1}{2n}(n_1, \ldots, n_k)\begin{pmatrix} 0 & \|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(2)}\|^2 & \cdots & \|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(k)}\|^2 \\ \|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(2)}\|^2 & 0 & \cdots & \|\bar{\mathbf{x}}_{(2)} - \bar{\mathbf{x}}_{(k)}\|^2 \\ & \vdots & & \\ \|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(k)}\|^2 & \|\bar{\mathbf{x}}_{(2)} - \bar{\mathbf{x}}_{(k)}\|^2 & \cdots & 0 \end{pmatrix}\begin{pmatrix} n_1 \\ \vdots \\ n_k \end{pmatrix} \\
&= \frac{1}{2n}\left(\sum_{s=1}^{k} n_s\|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(s)}\|^2, \ldots, \sum_{s=1}^{k} n_s\|\bar{\mathbf{x}}_{(k)} - \bar{\mathbf{x}}_{(s)}\|^2\right)\begin{pmatrix} n_1 \\ \vdots \\ n_k \end{pmatrix} \\
&= \frac{1}{2n}\sum_{r=1}^{k}\sum_{s=1}^{k} n_r n_s\|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_{(s)}\|^2 \\
&= \frac{1}{n}\sum_{1\le r<s\le k} n_r n_s\|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_{(s)}\|^2.
\end{aligned}$$

Obteniéndose el resultado deseado. □

Por tanto, el criterio de maximizar (3.3) equivale a maximizar

$$\frac{1}{n}\sum_{1\le r<s\le k} n_r n_s\|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_{(s)}\|^2. \tag{3.4}$$

Si se da el caso de que la distancia δ no es euclídea, tal como se ha comentado en el capítulo 2, es adecuado transformar la distancia antes de calcular las coordenadas principales. En particular se tiene,

Propiedad 3.3.

Dada un distancia δ no euclídea y utilizando la corrección $\delta_{ij}^{*^2} = \delta_{ij}^2 + h$ propuesta en el capítulo 2, minimizar $\sum_{r=1}^{k} n_r V_{\delta^*}(C_r)$ equivale a minimizar $\sum_{r=1}^{k} n_r V_{\delta}(C_r)$.

Demostración:

Es fácil ver que para las transformaciones afines $\delta_{ij}^{*^2} = a\delta_{ij}^2 + b$ $(\forall i, j)$ de los cuadrados de las distancias, las variabilidades geométricas correspondientes a δ y δ^* verifican $V_{\delta^*} = aV_{\delta} + \frac{b}{2}$. En particular, si $\delta_{ij}^{*^2} = \delta_{ij}^2 + h$, se tiene que

$$\sum_{r=1}^{k} n_r V_{\delta^*}(C_r) = \sum_{r=1}^{k} n_r V_{\delta}(C_r) + \frac{nh}{2},$$

lo que muestra la citada equivalencia. □

En todo el desarrollo de este método, se ha asumido que el número de clusters k está fijado desde el inicio por el investigador. Al aplicar un procedimiento divisivo, lo más común es tomar $k = 2$ e ir dividiendo cada cluster en dos nuevos clusters, repitiendo este procedimiento hasta que cada objeto forme un cluster o bien hasta que el investigador estime oportuno. Este procedimiento divisivo

proporciona una estructura de árbol que en nuestro caso no tiene porque tener una ultramétrica asociada (ver la nota final de la sección 3.3 de este mismo capítulo). De todas maneras, también cabe la posibilidad de estimar el número de clusters k que hay en los datos. Esta tarea no es evidente en absoluto y la estudiaremos con mayor profundidad en el capítulo 4 pero la clásica idea de mirar los cocientes de la variabilidad entre los clusters y dentro de los clusters nos lleva a proponer el siguiente criterio.

Criterio:

En el criterio divisivo propuesto, para cada valor de $k > 1$, se considera el cociente,

$$R_k = \frac{\sum_{r=1}^{k}\sum_{s=1}^{k} n_r n_s \Delta^2(C_r, C_s)/2n(k-1)}{\sum_{r=1}^{k} n_r V(C_r)/(n-k)}. \tag{3.5}$$

El valor de k que maximiza este cociente proporciona una estimación del número de clusters que hay en los datos.

Además se cumple la siguiente propiedad.

Propiedad 3.4.

Si los datos son datos continuos y si la distancia δ es la distancia euclídea, el criterio (3.4) se reduce al método politético y divisivo de Edwards and Cavalli-Sforza (1965) y el cociente R_k definido en (3.5) coincide con el propuesto en Calinski and Harabasz (1974).

Demostración:

Al considerar la distancia euclídea con datos continuos, las coordenadas principales coinciden con las componentes principales, es decir, son combinaciones lineales de las variables originales. Por tanto,

$$\mathbf{X} = \mathbf{Z}T,$$

donde T es la matriz de vectores propios obtenidos en la diagonalización de la matriz de covarianzas de las variables continuas iniciales, verificándose
$TT' = T'T = I$.

En Edwards and Cavalli-Sforza (1965) se propone ir dividiendo los datos en dos clusters C_1 y C_2 de manera que se maximice la variabilidad entre los clusters, concretamente maximizando

$$n_1 n_2 \|\bar{\mathbf{z}}_{(1)} - \bar{\mathbf{z}}_{(2)}\|^2.$$

Por tanto,

$$\begin{aligned} n_1 n_2 \|\bar{\mathbf{z}}_{(1)} - \bar{\mathbf{z}}_{(2)}\|^2 &= n_1 n_2 T(\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(2)})'(\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(2)})T' \\ &= n_1 n_2 TT' \|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(2)}\|^2 \\ &= n_1 n_2 \|\bar{\mathbf{x}}_{(1)} - \bar{\mathbf{x}}_{(2)}\|^2, \end{aligned}$$

lo que coincide con (3.4) tomando $k = 2$.

Calinski and Harabasz (1974) proponen estimar el número de clusters seleccionando aquella partición que maximiza el cociente de la variabilidad entre los

clusters y la variabilidad dentro de los clusters. Para ello definen el cociente,

$$CH_k = \frac{\sum_{r=1}^{k} n_r \|\bar{\mathbf{z}}_{(r)} - \bar{\mathbf{z}}\|^2 \,/(k-1)}{\sum_{r=1}^{k} \sum_{i \in C_r} \|\mathbf{z}_i - \bar{\mathbf{z}}_{(r)}\|^2 \,/(n-k)}\,, \tag{3.6}$$

donde $\bar{\mathbf{z}}_{(r)}$ representa la media del cluster C_r, $\mathbf{z}_i$ el i-ésimo objeto del cluster C_r, $(r = 1, \ldots, k)$ y $\bar{\mathbf{z}}$ la media de los n objetos, y estiman el número de clusters por el valor de k que maximiza CH_k.

Empezando por el numerador de R_k, ya se ha visto en (2.12) que $\Delta^2(C_r, C_s) = \|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_{(s)}\|^2$. Luego,

$$\begin{aligned}
\sum_{r=1}^{k}\sum_{s=1}^{k} n_r n_s \Delta^2(C_r, C_s) &= \sum_{r=1}^{k}\sum_{s=1}^{k} n_r n_s \|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_{(s)}\|^2 \\
&= \sum_{r=1}^{k}\sum_{s=1}^{k} n_r n_s \|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}} + \bar{\mathbf{x}} - \bar{\mathbf{x}}_{(s)}\|^2 \\
&= \sum_{r=1}^{k}\sum_{s=1}^{k} n_r n_s \left(\|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}\|^2 + (\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}})'(\bar{\mathbf{x}} - \bar{\mathbf{x}}_{(s)}) \right. \\
&\quad \left. + (\bar{\mathbf{x}} - \bar{\mathbf{x}}_{(r)})'(\bar{\mathbf{x}}_{(s)} - \bar{\mathbf{x}}) + \|\bar{\mathbf{x}} - \bar{\mathbf{x}}_{(s)}\|^2 \right) \\
&= 2n \sum_{r=1}^{k} n_r \|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}\|^2 \\
&= 2nT'T \sum_{r=1}^{k} n_r \|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}\|^2 \\
&= 2n \sum_{r=1}^{k} n_r \|\bar{\mathbf{z}}_{(r)} - \bar{\mathbf{z}}\|^2.
\end{aligned}$$

De manera similar, para el denominador,

$$\begin{aligned} n_r V(C_r) &= \frac{1}{2n_r} \sum_{i,j \in C_r} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \\ &= \frac{1}{2n_r} \sum_{i,j \in C_r} \|\mathbf{x}_i - \bar{\mathbf{x}}_{(r)} + \bar{\mathbf{x}}_{(r)} - \mathbf{x}_j\|^2 \\ &= \frac{1}{2n_r} \sum_{i,j \in C_r} \big(\|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_{(r)}\|^2 + (\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_{(r)})'(\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_j) \\ &\quad + (\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_j)'(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_{(r)}) + \|\bar{\mathbf{x}}_{(r)} - \bar{\mathbf{x}}_j\|^2 \big) \\ &= \sum_{i \in C_r} \|\mathbf{x}_i - \bar{\mathbf{x}}_{(r)}\|^2 \\ &= T'T \sum_{i \in C_r} \|\mathbf{x}_i - \bar{\mathbf{x}}_{(r)}\|^2 \\ &= \sum_{i \in C_r} \|\mathbf{z}_i - \bar{\mathbf{z}}_{(r)}\|^2. \end{aligned}$$

Por tanto, se obtiene el cociente deseado.□

Por otra parte es bien conocido que en los métodos divisivos encontrar la partición de C en k clusters presenta gran dificultad de cálculo. Por ejemplo, el número de diferentes modos de clasificar n objetos en $k = 2$ clusters es $2^{n-1}-1$. En cuanto n aumenta un poco, el problema resulta computacionalmente imposible. Pero el criterio de minimizar (3.2) también se puede abordar considerándolo como un problema de programación entera formulada de la siguiente manera.

Formulación como un problema de programación entera.

Se definen las variables binarias $x_{ir} \in \{0,1\}$, tomando el valor 1 si el objeto i pertenece al cluster C_r y tomando el valor 0 en otro caso, $(r = 1, ..., k)$. Definimos también las variables binarias z_{ijr} con valor 1 si el objeto i y el objeto j pertenecen al cluster C_r, y 0 en otro caso, $(r = 1, ..., k)$. De esta manera, minimizar (3.2) puede formularse como:

$$\text{Minimizar:} \quad \sum_{r=1}^{k} \left\{ \frac{1}{2n_r} \sum_{i,j} \delta_{ij}^2 z_{ijr} \right\}$$

$$\sum_{i} x_{ir} = n_r, \quad \forall r \in \{1, \cdots, k\} \tag{3.7}$$

$$\sum_{r=1}^{k} x_{ir} = 1, \quad \forall i \tag{3.8}$$

$$\sum_{j \neq i} z_{ijr} = (n_r - 1) x_{ir}, \quad \forall i, \quad \forall r \in \{1, \cdots, k\} \tag{3.9}$$

$$z_{ijr} \leq x_{ir}, \quad \forall i, j \quad \forall r \in \{1, \cdots, k\} \tag{3.10}$$

$$x_{ir} \in \{0,1\}, \quad \forall i, r; \quad z_{ijr} \in \{0,1\}, \quad \forall i, j, r \tag{3.11}$$

Las condiciones (3.7) y (3.8) aseguran que el cluster C_r tiene n_r objetos $(r = 1, ..., k)$ y que cada objeto se ha asignado a un único cluster C_r, $r = 1, \ldots, k$. Las condiciones (3.9) garantizan que tiene que haber exactamente $n_r - 1$ objetos junto con el objeto i en el cluster C_r, $r = 1, ..., k$. Finalmente, las condiciones (3.10) y (3.11) refuerzan esta formulación.

Para esta formulación de programación entera se ha escrito un programa eficiente para $n \leq 50$ y $k = 2$. Para finalizar, comentar que siempre se puede

aplicar una aproximación del tipo *hill-climbing*. Es decir, fijada cierta partición, se puede ir mejorando la partición reasignando cada uno de los objetos al cluster que ofrezca la mayor mejoría en el valor de la función objetivo. Esta reasignación termina cuando ninguna reasignación mejora el valor del criterio establecido.

3.2.2. GEVA - Algoritmos aglomerativos

Los procedimientos aglomerativos se inician con una partición $P_0 = \{\{\mathbf{z}_1\}, \ldots, \{\mathbf{z}_n\}\}$ donde cada objeto forma un cluster. Estos clusters se van uniendo dos a dos hasta que todos los objetos formen un único cluster y como es conocido, las diferencias entre los diferentes algoritmos radican en cómo se mide la distancia entre dos clusters y por tanto en la forma de unión de los clusters. A continuación se desarrollan tres algoritmos aglomerativos.

3.2.2.1. GEVA-Ward clustering

La expresión (3.2) se puede entender como una medida de la heterogenidad de la partición $P = \{C_r : \quad r = 1, \ldots, k\}$, y por tanto, desde el punto de vista de los algoritmos aglomerativos, es lógico unir los clusters C_p y C_q que produzcan el menor aumento de (3.2). Es decir, supongamos que en un paso determinado tenemos la partición $P = \{C_1, \ldots, C_p, \ldots, C_q, \ldots, C_k\}$, con el valor asociado

$$W(P) = \sum_{r=1}^{k} n_r V(C_r).$$

En el siguiente paso uniremos los clusters C_p y C_q obteniéndose la partición $P' = \{C_1\,, \ldots\,, C_p \cup C_q\,, \ldots\,, C_k\}$, si con esta unión el correspondiente valor de $W(P')$ es el mínimo posible.

Definición 3.1.

Se define la distancia entre dos clusters C_p y C_q como,

$$d(C_p, C_q) = (n_p + n_q)V(C_p \cup C_q) - n_pV(C_p) - n_qV(C_q).$$

Propiedad 3.5.

La distancia así definida tiene asociados los parámetros
$\alpha_1 = (n_p+n_s)/(n_p+n_q+n_s)$, $\alpha_2 = (n_q+n_s)/(n_p+n_q+n_s)$, $\beta = -n_s/(n_p+n_q+n_s)$ *y* $\gamma = 0$ *en la fórmula de recurrencia de Lance and Williams (1967).*

Demostración:

Lance and Williams (1967) agruparon muchos de los métodos aglomerativos en un único algoritmo aglomerativo general de manera que si en un paso dado se agrupaban los clusters C_p y C_q, las distancias entre el nuevo cluster $C_p \cup C_q$ y el resto de los clusters C_s se calculan mediante la siguiente fórmula:

$$d(C_p \cup C_q, C_s) = \alpha_1 d(C_p, C_s) + \alpha_2 d(C_q, C_s) + \beta d(C_p, C_q) + \gamma |d(C_p, C_s) - d(C_q, C_s)|.$$

Los diferentes valores de los 4 parámetros $(\alpha_1, \alpha_2, \beta, \gamma)$ de la fórmula recuperan algunos de los métodos clásicos (Everitt, 1993; Gordon, 1999).

Volviendo a nuestro caso particular y desarrollando la expresión $d(C_p \cup C_q, C_s)$, se tiene que:

$$\begin{aligned}
d(C_p \cup C_q, C_s) &= (n_p + n_q + n_s)V(C_p \cup C_q \cup C_s) - (n_p + n_q)V(C_p \cup C_q) - n_s V(C_s) \\
&= \frac{1}{n_p + n_q + n_s}(n_p + n_s)^2 V(C_p \cup C_s) + \frac{1}{n_p + n_q + n_s}(n_q + n_s)^2 V(C_q \cup C_s) \\
&+ \frac{1}{n_p + n_q + n_s}(n_p + n_q)^2 V(C_p \cup C_q) - \frac{1}{n_p + n_q + n_s} n_s^2 V(C_s) \\
&- \frac{1}{n_p + n_q + n_s} n_p^2 V(C_p) - \frac{1}{n_p + n_q + n_s} n_q^2 V(C_q) \\
&- (n_p + n_q)V(C_p \cup C_q) - n_s V(C_s);
\end{aligned}$$

de donde reordenando los términos:

$$\begin{aligned}
d(C_p \cup C_q, C_s) &= \frac{n_p + n_s}{n_p + n_q + n_s}\left[(n_p + n_s)V(C_p \cup C_s) - n_p V(C_p) - n_s V(C_s)\right] \\
&+ \frac{n_q + n_s}{n_p + n_q + n_s}\left[(n_q + n_s)V(C_q \cup C_s) - n_q V(C_q) - n_s V(C_s)\right] \\
&- \frac{n_s}{n_p + n_q + n_s}\left[(n_p + n_q)V(C_p \cup C_q) - n_p V(C_p) - n_q V(C_q)\right] \\
&= \frac{n_p + n_s}{n_p + n_q + n_s} d(C_p, C_s) + \frac{n_q + n_s}{n_p + n_q + n_s} d(C_q, C_s) + \frac{-n_s}{n_p + n_q + n_s} d(C_p, C_q). \square
\end{aligned}$$

Obsérvese que los valores de estos parámetros son los mismos que los del método clásico de Ward (Ward, 1963) y por tanto en el caso de que se tengan variables cuantitativas y se considere como distancia δ la distancia euclídea, este método coincide con el bien conocido método de Ward.

3.2.2.2. GEVA-Joining clustering

A continuación se propone un método aglomerativo en el que se buscan sucesivas particiones de manera que se maximice la cohesión interna de los clusters en términos de la variabilidad geométrica. Inicialmente se buscan los dos objetos que formen el cluster de menor variabilidad geométrica entre todos los pares de objetos. Evidentemente, esto es equivalente a seleccionar, de entre todos los pares de objetos, los dos objetos que estén a menor distancia. En los sucesivos pasos del algoritmo, se unirán los clusters C_p y C_q de manera que $V(C_p \cup C_q) = \text{mín}_{r,s}\{V(C_r \cup C_s)\}$. Este método conlleva la siguiente definición.

Definición 3.2.

Se define la distancia entre dos clusters C_p y C_q como

$$d(C_p, C_q) = V(C_p \cup C_q).$$

Nota: no es posible integrar este método en la fórmula de recurrencia de Lance and Williams y no es el caso más general de ningún otro método conocido.

Propiedad 3.6.

El algoritmo GEVA-joining tiene asociado una ultramétrica.

Demostración:

El método aglomerativo generará una ultramétrica (Jain and Dubes, 1988), si y sólo si, al unir los clusters C_p y C_q se tiene que,

$$d(C_p, C_q) \leq d(C_p \cup C_q, C_s), \quad C_s \neq C_p, C_q.$$

Para este método la desigualdad a comprobar es

$$V(C_p \cup C_q) \leq V(C_p \cup C_q \cup C_s).$$

Por una parte, el propio procedimiento del algoritmo lleva a las desigualdades

$$\begin{aligned}
&V(C_p) \leq V(C_p \cup C_s)\,; && V(C_q) \leq V(C_q \cup C_s)\,;\\
&V(C_s) \leq V(C_s \cup C_p)\,; && V(C_s) \leq V(C_s \cup C_q)\,;\\
&V(C_p \cup C_q) \leq V(C_p \cup C_s)\,; && V(C_p \cup C_q) \leq V(C_q \cup C_s)\,.
\end{aligned}$$

Así pues, desarrollando la expresión de variabilidad geométrica, se tiene que

$$\begin{aligned}
V(C_p \cup C_q \cup C_s) &= \frac{1}{(n_p+n_q+n_s)^2}\left[(n_s+n_p)^2 V(C_s \cup C_p) + (n_s+n_q)^2 V(C_s \cup C_q)\right.\\
&+ \left.(n_p+n_q)^2 V(C_p \cup C_q) - n_p^2 V(C_p) - n_q^2 V(C_q) - n_s^2 V(C_s)\right].
\end{aligned}$$

Por tanto, es fácil ver que la desigualdad a comprobar es equivalente a:

$$\begin{aligned}
&\left(n_s^2 + 2n_s n_p + 2n_s n_q\right) V(C_p \cup C_q) + n_s^2 V(C_s) + n_p^2 V(C_p) + n_q^2 V(C_q) \leq\\
&(n_s+n_p)^2 V(C_s \cup C_p) + (n_s+n_q)^2 V(C_s \cup C_q),
\end{aligned}$$

siendo evidente la comprobación de ésta teniendo en cuenta las citadas desigualdades derivadas directamente del procedimiento del algoritmo. □

3.2.2.3. GEVA-Centroid clustering

De forma análoga al algoritmo anterior, este algoritmo se inicia uniendo los dos objetos que están a menor distancia entre todos los pares de objetos. En este caso consideramos la siguiente distancia.

Definición 3.3.

La distancia entre dos clusters C_p y C_q viene dada por la distancia (al cuadrado) definida en (2.9), es decir,

$$d(C_p, C_q) = \Delta^2(C_p, C_q).$$

Proposición 3.1.

Este método se puede escribir según la fórmula de recurrencia de Lance and Williams con valores de los parámetros $\alpha_1 = n_p/(n_p + n_q)$, $\alpha_2 = n_q/(n_p + n_q)$, $\beta = -n_p n_q/(n_p + n_q)^2$ *y* $\gamma = 0$.

Demostración:

Supuesto que en cierto momento del algoritmo se unen los clusters C_p y C_q y

dado otro cluster cualquiera C_s, desarrollamos la expresión de:

$$
\begin{aligned}
d(C_p \cup C_q, C_s) &= \frac{1}{(n_p+n_q)n_s} \sum_{\substack{i \in C_p \cup C_q \\ j \in C_s}} \delta_{ij}^2 - V(C_p \cup C_q) - V(C_s) \\
&= \frac{1}{(n_p+n_q)n_s} \left(\sum_{\substack{i \in C_p \\ j \in C_s}} \delta_{ij}^2 + \sum_{\substack{i \in C_q \\ j \in C_s}} \delta_{ij}^2 \right) - \frac{1}{2(n_p+n_q)^2} \sum_{i,j \in C_p \cup C_q} \delta_{ij}^2 - V(C_s) \\
&= \frac{1}{(n_p+n_q)n_s} \left[n_p n_s (\Delta^2(C_p, C_s) + V(C_p) + V(C_s)) \right. \\
&+ \left. n_q n_s (\Delta^2(C_q, C_s) + V(C_q) + V(C_s)) \right] \\
&- \frac{1}{2(n_p+n_q)^2} \left(\sum_{i,j \in C_p} \delta_{ij}^2 + \sum_{i,j \in C_q} \delta_{ij}^2 + 2 \sum_{\substack{i \in C_p \\ j \in C_q}} \delta_{ij}^2 \right) - V(C_s) \\
&= \frac{1}{n_p+n_q} \left[n_p (\Delta^2(C_p, C_s) + V(C_p) + V(C_s)) + n_q (\Delta^2(C_q, C_s) + V(C_q) + V(C_s)) \right] \\
&- \frac{1}{2(n_p+n_q)^2} \left[2n_p^2 V(C_p) + 2n_q^2 V(C_q) \right. \\
&+ \left. 2n_p n_q (\Delta^2(C_p, C_q) + V(C_p) + V(C_q)) \right] - V(C_s) \\
&= \frac{n_p}{n_p+n_q} \Delta^2(C_p, C_s) + \frac{n_q}{n_p+n_q} \Delta^2(C_q, C_s) \\
&+ \frac{1}{n_p+n_q} (n_p V(C_p) + n_q V(C_q) + (n_p+n_q) V(C_s)) \\
&- \frac{1}{n_p+n_q} (n_p V(C_p) + n_q V(C_q)) - \frac{n_p n_q}{(n_p+n_q)^2} \Delta^2(C_p, C_q) - V(C_s) \\
&= \frac{n_p}{n_p+n_q} \Delta^2(C_p, C_s) + \frac{n_q}{n_p+n_q} \Delta^2(C_q, C_s) + \frac{-n_p n_q}{(n_p+n_q)^2} \Delta^2(C_p, C_q).
\end{aligned}
$$

Teniendo en cuenta cómo se ha definido en este caso la distancia entre clusters,

se tiene que:

$$
d(C_p \cup C_q, C_s) = \frac{n_p}{n_p+n_q} d(C_p, C_s) + \frac{n_q}{n_p+n_q} d(C_q, C_s) + \frac{-n_p n_q}{(n_p+n_q)^2} d(C_p, C_q). \square
$$

Cabe destacar que estos valores de los parámetros coinciden con los correspondientes del método clásico del centroide, de manera que cuando se parte de variables cuantitativas y se utiliza como distancia entre objetos la distancia euclídea, este método coincide con el método del centroide.

3.2.3. Algunas propiedades

Es bien sabido que tanto el método clásico de Ward como el del centroide tienden a formar clusters esféricos y que ninguno de los dos sufre del efecto de encadenamiento (Everitt, 1993). Sin embargo, en la literatura no hemos encontrado menciones de estas propiedades para el método de Cavalli-Sforza. Además, como el método GEVA-joining no recupera ningún método clásico, hemos querido ver cuál era el comportamiento respecto estos dos aspectos de los métodos GEVA-divisivo y GEVA-joining.

1. *¿Clusters elípticos o esféricos?.*

   Se dice que un método cluster tiene tendencia a formar clusters esféricos cuando aunque en realidad los objetos formen grupos con forma elíptica y estén bien diferenciados, el método tiende a no identificarlos y a encontrar clusters con forma esférica mezclándose los objetos de los dos grupos. La Figura 3.1 (izquierda) muestra un conjunto de objetos, los cuales se agrupan en dos clusters elípticos bien diferenciados. La Figura 3.1 (derecha) muestra los clusters que encontró el método de Ward observándose la tendencia de

éste método a encontrar clusters esféricos. Del total de objetos, 5 objetos del grupo C_1 (en negro) se juntan con 21 objetos del grupo C_2 (en negro) para formar un cluster. El resto de los objetos de los dos grupos (en cruz) se juntan en un segundo cluster.

Para estudiar el comportamiento de los métodos GEVA-divisivo y GEVA-joining en una situación como la que se acaba de comentar, se han simulado 10 conjuntos de datos cada uno con 2 clusters elípticos bien separados. Los tamaños de los clusters fueron aleatoriamente de 25 ó 40 objetos para la aplicación del GEVA-joining y de 7 ó 10 objetos para la aplicación del GEVA-divisivo. Los clusters se generaron a partir de dos normales bivariantes de medias $\mu_1 = (0\ 0)'$ y $\mu_2 = (4\ 4)'$, respectivamente y matriz de varianzas covarianzas común:

$$\Sigma = \begin{pmatrix} var1 & cov \\ cov & var2 \end{pmatrix},$$

donde $var1$ y $var2$ se eligieron al azar entre los valores $\{3^2, 4^2, 5^2\}$, y entre $\{0.25^2, 0.5^2, 0.75^2\}$, respectivamente. De manera similar, el coeficiente de correlación se seleccionó al azar entre los valores $\{0.2, 0.5, 0.8\}$ y a partir de éstos se calculó la covarianza cov. Como distancia inicial se tómo la distancia euclídea entre los objetos. No se eligió como distancia la distancia de Mahalanobis ya que su utilización hubiera eliminado el efecto de la correlación que propiamente hemos introducido al generar los datos.

Ni el GEVA-divisivo ni el GEVA-joining han identificado correctamente los dos clusters en ningúno de los conjuntos de objetos simulados.

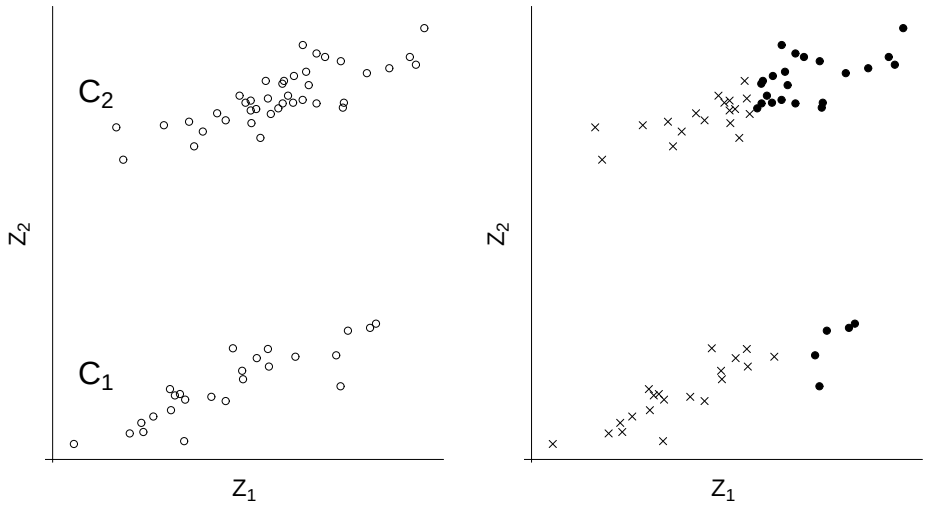


Figura 3.1: (Izquierda) Dos clusters elípticos bien diferenciados. (Derecha) Resolución que ofrece el método de Ward. Es evidente la tendencia del método a encontrar clusters esféricos.

2. *El efecto encadenamiento.*

   Este efecto se presenta cuando dos grupos de objetos están bien separados pero hay algunos pocos objetos entre los dos grupos que de alguna manera los conectan y como consecuencia el método de cluster no es capaz de identificar correctamente los dos grupos. A veces se refiere a este efecto como «los amigos de mis amigos son mis amigos». La Figura 3.2 muestra una situación de este tipo, entre dos grupos de objetos. Para recrear esta situación, hemos considerado conjuntos de objetos de forma que estos pertenecieran a dos grupos, cada uno de ellos formado por 15 objetos y generados a partir de dos distribuciones normales bivariantes de medias $\mu_1 = (0\ 0)'$ y $\mu_2 = (4\ 4)'$, respectivamente, y matriz de varianzas-covarianzas común e igual a la matriz identidad. Además, hemos añadido los objetos $(1.5, 1.5)$, $(2, 2)$ y $(2.5, 2.5)$ que entrelazan estos dos grupos. Hemos aplicado los métodos GEVA-divisivo y GEVA-joining usando como distancia inicial la distancia euclídea. Para ambos métodos los resultados indicaron que no sufrían del efecto encadenamiento.

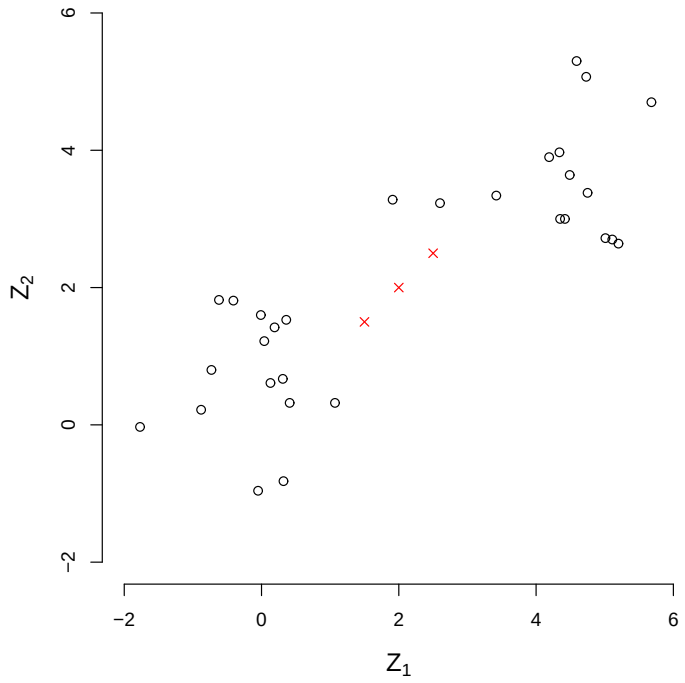


Figura 3.2: Representación de dos grupos de objetos bien separados pero unidos por 3 objetos (indicados por × en color rojo).

Para finalizar y antes de pasar a la aplicación de los métodos presentados a datos reales, se adjunta la Tabla 3.1 en la que se resumen las propiedades de los algoritmos GEVA y se comparan con las de algunos métodos clásicos.

Tabla 3.1: Propiedades de los métodos GEVA y comparación con algunos métodos clásicos

Método	Modo Jerárquico	Tipo Variables	Ultramétrica Asociada	Clusters Esféricos	Efecto Encadenamiento
Cavalli-Sforza	divisivo	cuantitativas	no	tendencia a	no
GEVA-divisivo	divisivo	todas	no	tendencia a	no
Ward	aglomerativo	cuantitativas	si	tendencia a	no
GEVA-Ward	aglomerativo	todas	si	tendencia a	no
Centroide	aglomerativo	cuantitativas	no	tendencia a	no
GEVA-centroide	aglomerativo	todas	no	tendencia a	no
GEVA-joining	aglomerativo	todas	si	tendencia a	no

3.3. Clasificación de poblaciones naturales de *Drosophila subobscura*

Para ilustrar los efectos de los diferentes algoritmos de clustering desarrollados en las secciones anteriores hemos considerado los datos derivados de un estudio de 40 muestras de polimorfismos para inversiones del cromosoma O de la especie *Drosophila subobscura* (Solé *et al.*, 2000; Balanyà *et al.*, 2004; Mestres *et al.*, 2008). El polimorfismo cromosómico para las inversiones es muy útil para caracterizar las muestras de las poblaciones naturales de la citada especie (Krimbas, 1993).

Tabla 3.2: Muestras de *Drosophila subobscura* etiquetadas de 1 a 40. La letra L entre paréntesis identifica las muestras letales.

Etiq.	Muestra	Etiq.	Muestra	Etiq.	Muestra	Etiq.	Muestra
1	Montpellier	11	Tübingen	21	Davis	31	Bordils(L)
2	Lagrasse	12	Vienna	22	Eureka	32	Kamariste(L)
3	Queralbs	13	Leuk	23	Medford	33	Petnica(L)
4	Riba-roja	14	Santiago Chile	24	Salem	34	Zanjic(L)
5	Calvià	15	Chillán	25	Centralia	35	Gilroy I(L)
6	Punta Umbría	16	Laja	26	Bellingham	36	Gilroy II(L)
7	Málaga	17	Valdivia	27	Port Hardy	37	Bellingham(L)
8	Groningen	18	Puerto Montt	28	Kamariste	38	Centralia(L)
9	Louvaine la Neuve	19	Coyhaique	29	Petnica	39	Santiago Chile(L)
10	Villars	20	Gilroy	30	Zanjic	40	Puerto Montt(L)

En la Tabla 3.2 se detallan las muestras de *Drosophila subobscura* incluidas en el estudio. Las muestras pueden pertenecer a cromosomas O letales (que como mínimo son portadoras de un gen letal recesivo) o de viabilidad normal. Estas muestras se corresponden a: europeas no letales (de 1 a 13 y de 28 a 30); europeas letales (de 31 a 34); americanas no letales (de 14 a 27) y finalmente muestras americanas letales (de 35 a 40). Un gen letal se define como aquel que produce la muerte en homocigosis, generalmente durante el desarrollo embrionario. El estudio de los genes letales es interesante en la genética evolutiva pues son unos indicadores de la variabilidad genética de las poblaciones y permiten estimar los parámetros que definen la estructura genética de dichas poblaciones.

Cada muestra P_i, $i = 1, ..., 40$, se carateriza por un vector m-dimensional $(p_{i1}, ..., p_{im})$ en el *espacio genético* cuyas coordenadas son frecuencias relativas de las inversiones para el cromosoma O, de manera que,

$$p_{ij} \geq 0,\ j = 1, ..., m \ \text{ y } \ \sum_{j=1}^{m} p_{ij} = 1; \quad i = 1, ..., 40.$$

En esta situación, cada objeto (en este caso, cada muestra) es en realidad el resultado de la agrupación de varios casos (en este caso, individuos de la misma especie) y es precisamente en este tipo de situaciones donde es adecuado utilizar la distancia de Bhattacharyya (ver (2.1) del capítulo 2). Por tanto, hemos considerado la distancia entre dos muestras $P_i = (p_{i1}, ..., p_{im})$ y $P_j = (p_{j1}, ..., p_{jm})$ como

$$d(P_i, P_j) = \arccos \sum_{l=1}^{m} \sqrt{p_{il} p_{jl}}.$$

Sobre este conjunto de datos se han aplicado los cuatro métodos de clus-

ter descritos en este capítulo. En las Figuras 3.3, 3.4 y 3.5 (páginas 62, 63 y 64, respectivamente) se muestran gráficamente los resultados obtenidos con los métodos GEVA-Ward, GEVA-joining y GEVA-divisivo, respectivamente. El método GEVA-centroide no ofrece en este caso una buena clasificación por lo que no lo consideraremos en esta discusión. A continuación pasamos a comentar los resultados obtenidos dentro del contexto del origen de los datos.

En primer lugar, comentar que los tres métodos de clustering ofrecen diferentes clasificaciones pero que todos tienen en común la división entre las muestras europeas y americanas. Esto es correcto ya que el polimorfismo cromosómico de las muestras americanas es escaso debido a que proceden de un proceso colonizador reciente en el que se redujo drásticamente la variabilidad genética (Prevosti *et al.*, 1985; Ayala *et al.*, 1989; Mestres *et al.*, 2005). Los métodos GEVA-Ward y GEVA-divisivo diferencian las muestras de los Balcanes del resto de las muestras europeas, sin embargo, el GEVA-joining no muestra tal diferencia. La composición cromosómica de las muestras de los Balcanes es particular (Zivanovic *et al.*, 2000) por lo que es lógica la diferenciación que hacen los dos primeros métodos. Sin embargo GEVA-joining señala en Europa un primer cluster con algunas muestras del Mediterráneo (4, 7 y 6) mientras que otras de la misma región se agrupan aparte (por ejemplo la muestra 5). GEVA-joining sólo diferencia las muestras de los Balcanes para un mayor valor con punto de corte en el dendrograma. Un segundo punto a destacar es que tanto GEVA-Ward como GEVA-divisivo encuentran un cluster de muestras ibero-mediterráneas. La diferencia entre los dos métodos consiste en que GEVA-divisivo agrupa todas las

muestras ibero-mediterráneas (3, 4, 5, 6 y 7) mientras que GEVA-Ward clasifica la muestra 3 junto con muestras de Francia y la muestra letal de Bordils (Gerona). Para acabar respecto a las muestras europeas, queda comentar que GEVA-divisivo agrupa a las muestras francesas y belgas junto con la muestra letal de Bordils (31, 9, 10, 1 y 2) de una manera bastante lógica, mientras que GEVA-Ward forma un cluster con las muestras francesas más la muestra letal de Bordils y la muestra de Queralbs (Gerona) y otro cluster con el resto de muestras europeas, como era esperable. Sin embargo, GEVA-joining no agrupa bien el resto de la muestras europeas, que se encuentran más bien entremezcladas.

Respecto a las muestras americanas, tanto GEVA-Ward como GEVA-divisivo detectan el cluster de muestras letales (de 35 a 40), mientras que GEVA-joining no lo hace. Cabe recordar que en las poblaciones americanas el polimorfismo cromosómico para las inversiones de las muestras letales difiere significativamente del de las no letales (Mestres *et al.*, 2005). Ello es debido a que en América se encuentran asociaciones entre inversiones y genes letales producidas por el fuerte efecto fundador de la reciente colonización (Mestres *et al.*, 1995). Probablemente, los clusters obtenidos con las muestras letales americanas están un poco mejor resueltos con el GEVA-divisivo que con el GEVA-Ward. El resto de las muestras americanas están correctamente clasificadas con GEVA-joining y GEVA-Ward aunque parece que GEVA-divisivo da una clasificación más precisa.

Estos resultados vuelven a mostrar que las clasificaciones que se obtienen dependen crucialmente del algoritmo aplicado. También reflejan que las clasificaciones de las 40 muestras obtenidas con GEVA-divisivo y GEVA-Ward son tan

realistas como se esperaba atendiendo a la información genética que se tiene de ellas.

Nota: Ya se ha comentado en el desarrollo del método GEVA-divisivo que éste en general no ofrece una jerarquía indexada o ultramétrica. Cabe destacar en este sentido que los clusters $C = \{35, 36, 37, 38, 39, 40\}$ y $C' = \{14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27\}$ se unen de manera que $V(C \cup C') = 0.170$ pero sin embargo $V(C) = 0.280 > V(C \cup C')$.

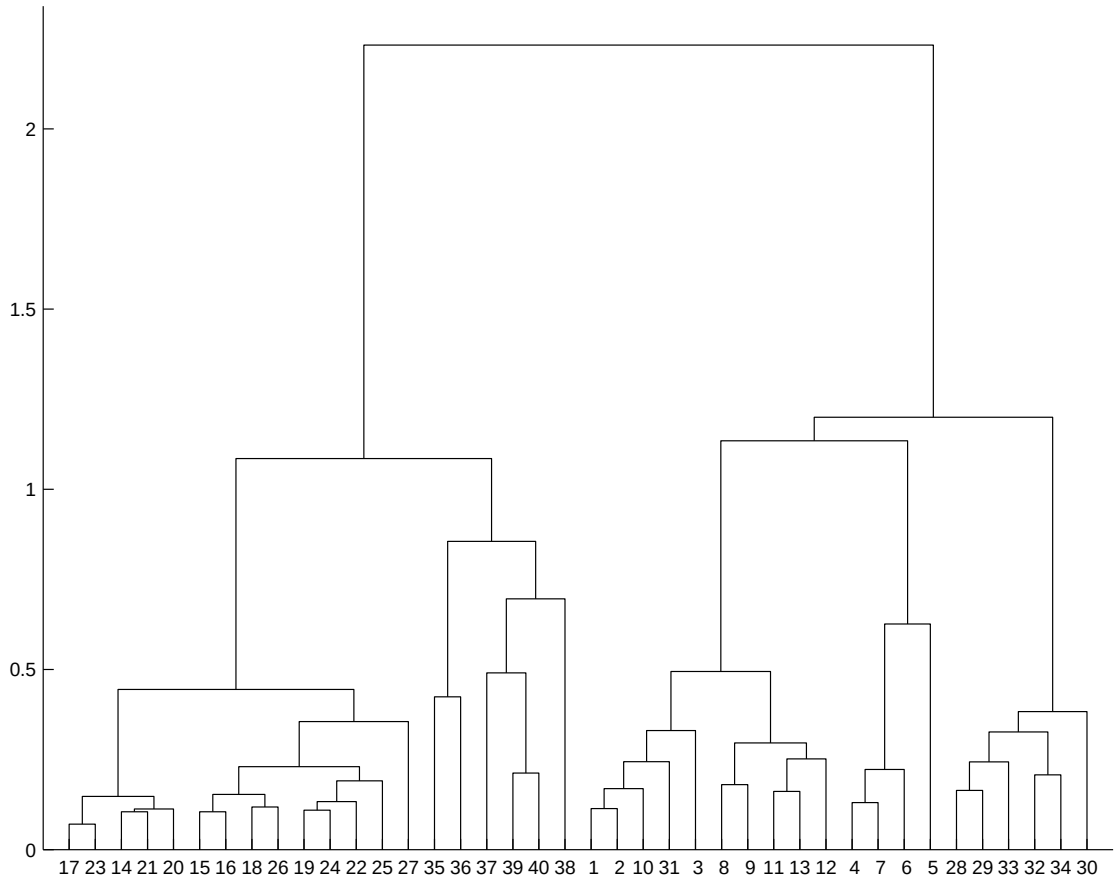


Figura 3.3: Dendrograma asociado al método GEVA-Ward aplicado a los datos del polimorfismo cromosómico de *Drosophila subobscura*.

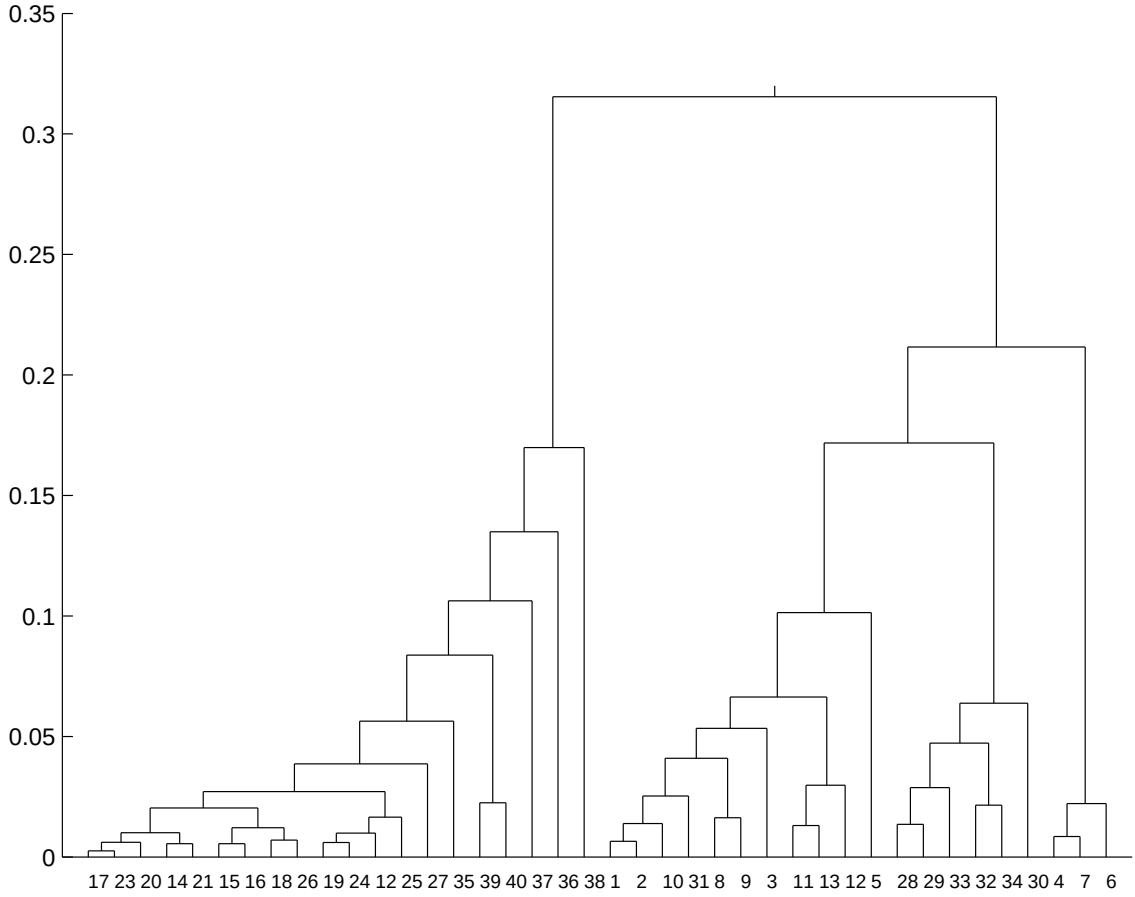


Figura 3.4: Dendrograma asociado al método GEVA-joining aplicado a los datos del polimorfismo cromosómico de *Drosophila subobscura*.

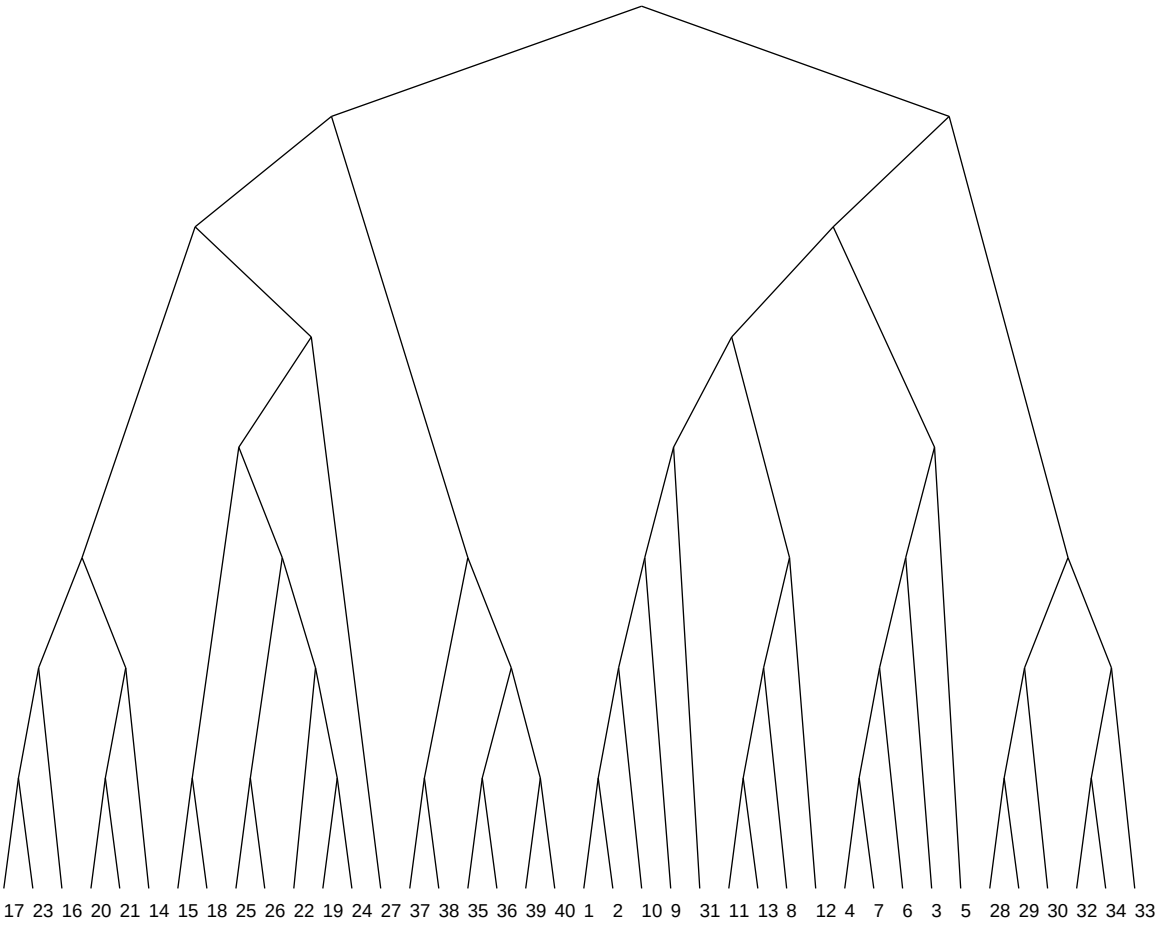


Figura 3.5: Árbol asociado al método GEVA-divisivo aplicado a los datos del polimorfismo cromosómico de *Drosophila subobscura*.

3.4. Conclusiones

En este capítulo de la memoria se han presentado algunos métodos nuevos de cluster basados en el concepto de variabilidad geométrica. Estos métodos son muy generales en el sentido que sólo necesitan las distancias entre los objetos para poder ser aplicados. Los métodos GEVA propuestos se pueden aplicar a datos binarios, cualitativos o mixtos y además ofrecen un marco teórico con una interpretración sencilla e intuitiva para ciertos métodos de cluster clásicos cuando se requiere aplicarlos a datos mixtos. Por otra parte, también se han obtenido buenos resultados en la aplicación a la clasificación de muestras de *Drosophila subobscura* respecto a su polimorfismo cromosómico para las inversiones.

3.5. Difusión del método

Una primera aproximación del método divisivo se presentó a la comunidad científica en la *IX Conferencia de Biometría*, 2003, celebrada en A Coruña. Todos los resultados presentados en este capítulo de la memoria forman parte de un artículo, titulado «GEVA: Geometric Variability-Based Approaches For Identifying Patterns On Data» que actualmente está sometido para su publicación.

3.6. Software desarrollado

El paquete desarrollado `DBmethods` contiene las siguientes funciones para poder ejecutar los métodos presentados:

- `dbhatta(x)`: dada una matriz de datos `x` de manera que cada fila contiene las frecuencias observadas de una distribución multinomial, calcula la matriz de distancias de Bhattacharyya descrita en el capítulo 2 (Anexo A, pág. 229; Anexo B, pág. 270).

- `deltas(d, pert)`: dadas una matriz de distancias `d` correspondiente a los objetos de una partición en clusters de un conjunto C y la composición de dicha partición, `pert`, calcula las distancias (al cuadrado) entre todos los pares de clusters organizadas en una matriz cuadrada (Anexo A, pág. 232; Anexo B, pág. 272).

- `GEVAdivisive(d)`: dada una matriz de distancias `d`, calcula la partición en dos clusters según el método divisivo GEVA-divisive. Hay que destacar que el tiempo de ejecución puede ser muy elevado.

  Esta función no desarrolla todo el proceso divisivo pero se ha construido en *matlab* un programa que sí lleva a cabo todas las sucesivas divisiones. Por diferencias en el tratamiento de punteros que hace R, en esta versión del paquete `DBmethods` no se ha incluído una función que realice complemtamente el proceso divisivo. De todas maneras se tiene previsto construir en R una función que desarrolle todo el proceso divisivo y así ir mejorando y

ampliando el paquete (Anexo A, pág. 221; Anexo B, pág. 284).

- `GEVAjoining(d)`: dada una matriz de distancias `d`, desarrolla todo el proceso de aglomeración del método de cluster GEVA-joining (Anexo A, pág. 223; Anexo B, pág. 286). Esta función está basada en la función `hierclust` de F. Murtagh y necesita la función auxiliar `getnns` (Anexo B, pág. 283).

- `proxi(d, dx0, pert= "onegroup")`: dadas una matriz de distancias `d` correspondiente a los objetos de una partición en clusters de un conjunto C, las distancias `dx0` de un nuevo objeto $\mathbf{z}_0$ a los objetos de C y la composición de dicha partición `pert`, calcula la función proximidad (al cuadrado) de $\mathbf{z}_0$ a cada cluster. Por defecto, la función asume que todos los objetos de C forman un único cluster y calcula la función proximidad (al cuadrado) de $\mathbf{z}_0$ a C (Anexo A, pág. 245; Anexo B, pág. 309).

- `vgeo(d, pert= "onegroup")`: dadas una matriz de distancias `d` correspondiente a los objetos de una partición en clusters de un conjunto C y la composición de dicha partición, `pert`, calcula la variabilidad geométrica de cada cluster. Por defecto, la función asume que todos los objetos forman un único cluster y calcula la variabilidad geométrica del conjunto C (Anexo A, pág. 251; Anexo B, pág. 316).

En este capítulo se han descrito los métodos de cluster GEVA-Ward y GEVA-centroide. El paquete `DBmethods` no incluye ninguna función para su cálculo ya que el paquete `stats` de R contiene la función `hclust` que permite su cálculo.

Capítulo 4

El número de clusters y los objetos atípicos

En este capítulo se presenta un nuevo estadístico, que denominaremos INCA (Index Number Clusters Atypical), a fin de resolver dos problemas que se prensentan en la clasificación de objetos. Por una parte, el problema de la estimación del número de clusters en un conjunto de n objetos, y por otra, el problema de la tipicalidad o el de decidir si un nuevo objeto pertenece a los clusters encontrados o si es un objeto atípico, es decir, si pertenece a algún otro grupo no considerado. Para ilustrar esta nueva técnica, se realizan diferentes estudios de simulación y se aplica sobre dos conjuntos de datos reales. Como es habitual en la memoria, el capítulo finaliza con una descripción del programa creado para la fácil aplicación de estos métodos.

4.1. Estadístico INCA

Sea un conjunto de n objetos en el que se presupone una estructura de clusters. Supongamos pues, que los n objetos están clasificados en k clusters $C_1, \ldots, C_k$, de tamaños $n_1, \ldots, n_k$, respectivamente y que estos clusters vienen representados por k vectores aleatorios independientes $\mathbf{Z}_1, \ldots, \mathbf{Z}_k$ con valores en un espacio métrico $\mathcal{R}$, con función de densidad $f_1, \ldots, f_k$ respecto de una medida adecuada y común λ. Sea δ una función distancia sobre $\mathcal{R}$. Fijemos un objeto $\mathbf{z}_0$ que en un principio puede ser tanto un objeto perteneciente a algún cluster C_l, $(l = 1, \ldots, k)$ o puede ser un objeto que pertenezca a algún cluster desconocido, es decir, un objeto atípico. Se considera el nuevo cluster con δ-media (ver capítulo 2) la combinación lineal $\sum_{r=1}^{k} \alpha_r E(\psi(\mathbf{Z}_r))$, donde como en el capítulo anterior, $\psi : \mathcal{R} \longrightarrow \mathbf{R}^q$ representa la función que proporciona las coordenadas principales.

Se define el estadístico INCA de la siguiente manera:

Definición 4.1.

En el contexto de una estructura de k clusters como la que se acaba de fijar y dado un nuevo objeto $\mathbf{z}_0$*, se define el estadístico* INCA *como:*

$$W(\mathbf{z}_0) = \min_{\alpha_r} \{L(\mathbf{z}_0)\}, \qquad \sum_{r=1}^{k} \alpha_r = 1 \tag{4.1}$$

donde

$$L(\mathbf{z}_0) = \sum_{r=1}^{k} \alpha_r \phi^2(\mathbf{z}_0, C_r) \; - \sum_{1 \leq r < s \leq k} \alpha_r \alpha_s \Delta^2(C_r, Cs). \tag{4.2}$$

Nota: en lo que sigue y para agilizar la notación denotaremos
$\phi^2(\mathbf{z}_0, C_r) = \phi_r^2(\mathbf{z_0})$ y $\Delta^2(C_r, C_s) = \Delta_{rs}^2$ $(r, s = 1, \ldots, k)$.

Teniendo en cuenta que $\phi_r^2(\mathbf{z}_0)$ es la función de proximidad de $\mathbf{z}_0$ al cluster C_r y que Δ_{rs}^2 es la distancia (al cuadrado) entre los clusters C_r y C_s, el estadístico INCA pretende minimizar la suma ponderada de las proximidades de $\mathbf{z}_0$ a los clusters (cosa que tiene en consideración la variabilidad dentro de los clusters), a la vez que maximiza la suma ponderada de distancias (al cuadrado) entre clusters (variabilidad entre clusters), lo que es un criterio muy común en el contexto del cluster.

La solución de (4.1) se puede hallar mediante los multiplicadores de Lagrange y se obtiene a partir del siguiente conjunto, $r = 1, ..., k-1$, de ecuaciones:

$$-(\Delta_{rk}^2 + \phi_k^2(\mathbf{z}_0) - \phi_r^2(\mathbf{z}_0)) + 2\alpha_r\Delta_{rk}^2 + \sum_{m=1, m\neq r}^{k-1} \alpha_m(\Delta_{rk}^2 + \Delta_{mk}^2 - \Delta_{rm}^2) = 0.$$

Por consiguiente, si M y N denotan las matrices

$$M = \begin{pmatrix} 2\Delta_{1k}^2 & \Delta_{1k}^2 + \Delta_{2k}^2 - \Delta_{12}^2 & \cdots & \Delta_{1k}^2 + \Delta_{(k-1)k}^2 - \Delta_{1(k-1)}^2 \\ \vdots & 2\Delta_{2k}^2 & & \vdots \\ \Delta_{1k}^2 + \Delta_{(k-1)k}^2 - \Delta_{1(k-1)}^2 & \cdots\cdots\cdots\cdots\cdots & & 2\Delta_{(k-1)k}^2 \end{pmatrix},$$

$$N = \begin{pmatrix} \Delta_{1k}^2 + \phi_k^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0) \\ \vdots \\ \Delta_{(k-1)k}^2 + \phi_k^2(\mathbf{z}_0) - \phi_{(k-1)}^2(\mathbf{z}_0) \end{pmatrix}, \tag{4.3}$$

los valores de $\alpha' = (\alpha_1, \ldots, \alpha_{k-1})$, junto con $\alpha_k = 1 - \sum_{r=1}^{k-1} \alpha_r$, que verifican (4.1) son $\alpha = M^{-1}N$.

El estadístico INCA junto con los estadísticos $U_r(\mathbf{z}_0) = \phi_r^2(\mathbf{z}_0) - W(\mathbf{z}_0)$, $(r = 1, \ldots, k)$ tienen una interesante interpretación geométrica que se pasa a detallar a continuación, así como su relación con algún estadístico ya conocido.

Proposición 4.1.

Sean $\mathbf{a}_r$ *la* δ*-media de* C_r, $r = 1, \ldots, k$, *y consideremos el punto* $\mathbf{z}_0$ *junto con el hiperplano generado por* $\mathbf{a}_1, \ldots, \mathbf{a}_k$. *Entonces,* $W(\mathbf{z}_0)$ *es la distancia (al cuadrado) o altura (al cuadrado) de* $\mathbf{z}_0$ *al hiperplano* $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$ *y* $U_r(\mathbf{z}_0)$, *es la proyección (al cuadrado) de la arista* $\{\mathbf{z}_0, \mathbf{a}_r\}$, $r = 1, \ldots, k$, *sobre el hiperplano* $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$.

Demostración:

Por simplicidad presentaremos únicamente la demostración para el caso $k = 3$. Consideramos pues el tetraedro $\{\mathbf{z}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ con $\mathbf{u}_r = E(\psi(\mathbf{Z}_r)) - \psi(\mathbf{z}_0)$, $r = 1, 2, 3$, donde $\mathbf{Z}_r$ es el vector aleatorio que representa al cluster C_r y ψ es la función que proporcionan las coordenadas principales (ver Figura 4.1). En base a (2.6) y (2.7) del capítulo 2, se tienen las siguiente igualdades $\mathbf{u}_1 \cdot \mathbf{u}_1 = \phi_1^2(\mathbf{z}_0)$, $\mathbf{u}_2 \cdot \mathbf{u}_2 = \phi_2^2(\mathbf{z}_0)$, $\mathbf{u}_3 \cdot \mathbf{u}_3 = \phi_3^2(\mathbf{z}_0)$, y además se verifica que $\|E(\psi(\mathbf{Z}_r)) - E(\psi(\mathbf{Z}_s))\|^2 = \Delta_{rs}^2$ $(r, s = 1, 2, 3)$. Esto, junto con el teorema del coseno lleva a,

$$\begin{aligned}
\mathbf{u}_1 \cdot \mathbf{u}_2 &= (\phi_1^2(\mathbf{z}_0) + \phi_2^2(\mathbf{z}_0) - \Delta_{12}^2)/2, \\
\mathbf{u}_2 \cdot \mathbf{u}_3 &= (\phi_2^2(\mathbf{z}_0) + \phi_3^2(\mathbf{z}_0) - \Delta_{23}^2)/2, \\
\mathbf{u}_1 \cdot \mathbf{u}_3 &= (\phi_1^2(\mathbf{z}_0) + \phi_3^2(\mathbf{z}_0) - \Delta_{13}^2)/2.
\end{aligned}$$

Como el volumen del paralelepípedo determinado por el producto mixto de

$\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ es seis veces el volumen V del tetraedro, es fácil ver que:

$$\begin{aligned} 144V^2 &= \phi_1^2(\mathbf{z}_0)\Delta_{23}^2(\Delta_{12}^2+\Delta_{13}^2-\Delta_{23}^2+\phi_2^2(\mathbf{z}_0)+\phi_3^2(\mathbf{z}_0)-\phi_1^2(\mathbf{z}_0)) \\ &+ \phi_2^2(\mathbf{z}_0)\Delta_{13}^2(\Delta_{23}^2+\Delta_{12}^2-\Delta_{13}^2+\phi_1^2(\mathbf{z}_0)+\phi_3^2(\mathbf{z}_0)-\phi_2^2(\mathbf{z}_0)) \\ &+ \phi_3^2(\mathbf{z}_0)\Delta_{12}^2(\Delta_{23}^2+\Delta_{13}^2-\Delta_{12}^2+\phi_1^2(\mathbf{z}_0)+\phi_2^2(\mathbf{z}_0)-\phi_3^2(\mathbf{z}_0)) \\ &- \Delta_{12}^2\Delta_{23}^2\Delta_{13}^2-\phi_1^2(\mathbf{z}_0)\phi_2^2(\mathbf{z}_0)\Delta_{12}^2-\phi_2^2(\mathbf{z}_0)\phi_3^2(\mathbf{z}_0)\Delta_{23}^2-\phi_1^2(\mathbf{z}_0)\phi_3^2(\mathbf{z}_0)\Delta_{13}^2. \end{aligned}$$

En lo que sigue, A designará la parte derecha de la ecuación anterior.

Por otra parte, puesto que el volumen del tetraedro es 1/3 del área de la base por la altura, tenemos que:

$$144V^2 = h^2(\Delta_{12}+\Delta_{23}+\Delta_{13})(\Delta_{12}+\Delta_{23}-\Delta_{13})(\Delta_{12}+\Delta_{13}-\Delta_{23})(\Delta_{23}+\Delta_{13}-\Delta_{12}).$$

Luego, la altura (al cuadrado) h^2 es:

$$\frac{A}{(\Delta_{12}+\Delta_{23}+\Delta_{13})(\Delta_{12}+\Delta_{23}-\Delta_{13})(\Delta_{12}+\Delta_{13}-\Delta_{23})(\Delta_{23}+\Delta_{13}-\Delta_{12})}. \tag{4.4}$$

Nótese que el denominador de (4.4) coincide con $|M|$, siendo M la matriz dada en (4.3), ya que

$$|M| = -(\Delta_{12}^4+\Delta_{23}^4+\Delta_{13}^4)+2(\Delta_{12}^2\Delta_{23}^2+\Delta_{12}^2\Delta_{13}^2+\Delta_{13}^2\Delta_{23}^2).$$

Teniendo en cuenta que $\alpha_3 = 1-\alpha_1-\alpha_2$ y que por tanto $W(\mathbf{z}_0)$ se escribe

como:

$$\begin{aligned} W(\mathbf{z}_0) &= \left\{\phi_3^2(\mathbf{z}_0) - \alpha_1(\Delta_{13}^2 + \phi_3^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0)) - \alpha_2(\Delta_{23}^2 + \phi_3^2(\mathbf{z}_0) - \phi_2^2(\mathbf{z}_0))\right. \\ &+ \left. \alpha_1^2\Delta_{13}^2 + \alpha_2^2\Delta_{23}^2 + \alpha_1\alpha_2(\Delta_{13}^2 + \Delta_{23}^2 - \Delta_{12}^2)\right\}, \end{aligned}$$

con

$$\alpha_1 = \frac{1}{|M|}\left(2(\Delta_{13}^2 + \phi_3^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0))\Delta_{23}^2 - (\Delta_{23}^2 + \phi_3^2(\mathbf{z}_0) - \phi_2^2(\mathbf{z}_0))(\Delta_{13}^2 + \Delta_{23}^2 - \Delta_{12}^2)\right),$$

$$\alpha_2 = \frac{1}{|M|}\left(2(\Delta_{23}^2 + \phi_3^2(\mathbf{z}_0) - \phi_2^2(\mathbf{z}_0))\Delta_{13}^2 - (\Delta_{13}^2 + \phi_3^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0))(\Delta_{13}^2 + \Delta_{23}^2 - \Delta_{12}^2)\right),$$

es sencillo ver mediante un pequeño cálculo que $W(\mathbf{z}_0)$ coincide con (4.4), y por tanto con h^2. $\square$

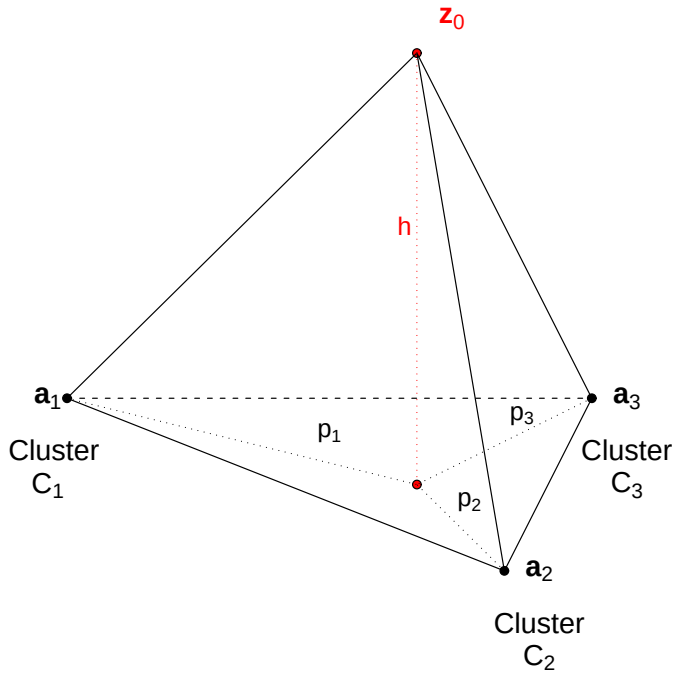


Figura 4.1: Representación gráfica de la proposición 4.1 para el caso $k = 3$. Los centros de los clusters vienen representados por los puntos $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ y las proyecciones (al cuadrado) de las aristas $\{\mathbf{z}_0, \mathbf{a}_l\}$ sobre el plano $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ vienen representadas por p_r, $r = 1, 2, 3$. La altura h (al cuadrado) representa el valor de $W(\mathbf{z}_0)$.

De esta interpretación del estadístico INCA como la distancia o la altura h (al cuadrado) desde $\mathbf{z}_0$ al hiperplano generado por las δ-medias de C_r $(r = 1, \ldots, k)$, de forma natural podemos decir que aquellos puntos que se encuentran significativamente lejos de este hiperplano serán objetos atípicos. Esta idea intuitiva es la que hemos utilizado tanto para la estimación del número de clusters como para la detección de objetos atípicos respecto de los clusters considerados. Obsérvese que cuando el número k de clusters excede la dimensionalidad p, el hiperplano generado por los centros de los clusters será todo el espacio de manera que el valor del estadístico INCA será siempre cero. En efecto,

Proposición 4.2.

En el contexto de una estructura de k clusters como la que se ha fijado al inicio de esta sección y dado un nuevo objeto $\mathbf{z}_0$*, sea* δ *una función de distancia euclídea con* $\psi : \mathcal{R} \longrightarrow \mathbf{R}^q$ *la función que proporciona las coordenadas principales. Entonces, se tiene que:*

1. *El estadístico INCA es mayor o igual a 0,* i.e.*,* $W(\mathbf{z}_0) \geq 0$.

2. *Si el número de clusters k es mayor que la dimensionalidad q de los datos, entonces el estadístico INCA es nulo.*

3. *Sean* $E(\psi(\mathbf{Z}_r))$ $(r = 1, \ldots, k)$ *las* δ*-medias de* C_r $(r = 1, \ldots, k)$*. Si* $\psi(\mathbf{z}_0)$ *es co-lineal con* $E(\psi(\mathbf{Z}_1)), \ldots, E(\psi(\mathbf{Z}_k))$*, entonces,* $W(\mathbf{z}_0) = 0$.

Demostración:

Nota: Para la agilizar la lectura de la demostración identificaremos los valores de las δ-medias de C_r, $E(\psi(\mathbf{Z}_r))$, por $\mathbf{a}_r$ $(r = 1, \ldots, k)$.

La demostración de la proposición radica en ver que

$$L(\mathbf{z}_0) = \|\psi(\mathbf{z}_0) - \sum_{r=1}^{k} \alpha_r \mathbf{a}_r\|^2, \quad \text{con} \quad \sum_{r=1}^{k} \alpha_r = 1. \tag{4.5}$$

Para ello recordemos que según (2.6) y (2.7) $\phi_r^2(\mathbf{z}_0) = \|\psi(\mathbf{z}_0) - \mathbf{a}_r\|^2$ y $\Delta_{rs}^2 = \|\mathbf{a}_r - \mathbf{a}_s\|^2$, $r, s = 1, \ldots, k$. Luego,

$$L(\mathbf{z}_0) = \sum_{r=1}^{k} \alpha_r \|\psi(\mathbf{z}_0) - \mathbf{a}_r\|^2 - \sum_{1 \le r < s \le k} \alpha_r \alpha_s \|\mathbf{a}_r - \mathbf{a}_s\|^2.$$

Sea $\mathbf{a} = \sum_{r=1}^{k} \alpha_r \mathbf{a}_r$, entonces,

$$\begin{aligned} L(\mathbf{z}_0) &= \sum_{r=1}^{k} \alpha_r \|\psi(\mathbf{z}_0) - \mathbf{a} + \mathbf{a} - \mathbf{a}_r\|^2 - \sum_{1 \le r < s \le k} \alpha_r \alpha_s \|\mathbf{a}_r - \mathbf{a}_s\|^2 \\ &= \sum_{r=1}^{k} \alpha_r \|\psi(\mathbf{z}_0) - \mathbf{a}\|^2 + \underbrace{\sum_{r=1}^{k} \alpha_r \|\mathbf{a} - \mathbf{a}_r\|^2}_{T_1} \\ &+ \underbrace{\sum_{r=1}^{k} \alpha_r (\psi(\mathbf{z}_0) - \mathbf{a})'(\mathbf{a} - \mathbf{a}_r)}_{T_2} - \underbrace{\sum_{1 \le r < s \le k} \alpha_r \alpha_s \|\mathbf{a}_r - \mathbf{a}_s\|^2}_{T_3}. \end{aligned}$$

Es claro que $T_2 = 0$ y por tanto, basta ver que $T_1 = T_3$ para demostrar (4.5).

Recordando que $\mathbf{a} = \sum_{s=1}^{k} \alpha_s \mathbf{a}_s$ y $\sum_{s}^{k} \alpha_s = 1$,

$$\begin{aligned} T_1 &= \sum_{r=1}^{k} \alpha_r \|\mathbf{a} - \mathbf{a}_r\|^2 \\ &= \sum_{r=1}^{k} \alpha_r \| \sum_{s=1}^{k} \alpha_s \mathbf{a}_s - \mathbf{a}_r\|^2 \\ &= \sum_{r=1}^{k} \alpha_r \sum_{\substack{s=1 \\ s\neq r}}^{k} \alpha_s^2 \|\mathbf{a}_s - \mathbf{a}_r\|^2 \\ &+ \sum_{r=1}^{k} \alpha_r \left[\sum_{s\neq l} \alpha_s \alpha_l \left\{ (\mathbf{a}_s - \mathbf{a}_r)'(\mathbf{a}_l - \mathbf{a}_r) + (\mathbf{a}_l - \mathbf{a}_r)'(\mathbf{a}_s - \mathbf{a}_r) \right\} \right]. \end{aligned}$$

Reordenando los términos de cada uno de los sumandos,

$$\begin{aligned} T_1 &= \sum_{r<s} \alpha_r \alpha_s \left[\alpha_r \|\mathbf{a}_r - \mathbf{a}_s\|^2 + \alpha_s \|\mathbf{a}_r - \mathbf{a}_s\|^2 \right] \\ &+ \sum_{r<s<l} \alpha_r \alpha_s \alpha_l \left[\|\mathbf{a}_r - \mathbf{a}_s\|^2 + \|\mathbf{a}_r - \mathbf{a}_l\|^2 + \|\mathbf{a}_s - \mathbf{a}_l\|^2 \right]. \end{aligned}$$

Por último, sacando factores comunes,

$$\begin{aligned} T_1 &= \sum_{r<s} \alpha_r \alpha_s \left[\alpha_r \|\mathbf{a}_r - \mathbf{a}_s\|^2 + \alpha_s \|\mathbf{a}_r - \mathbf{a}_s\|^2 \right] \\ &+ \sum_{r<s} \alpha_r \alpha_s \left[\sum_{\substack{l \\ l\neq r,s}} \alpha_l \|\mathbf{a}_r - \mathbf{a}_s\|^2 \right] \\ &= \sum_{r<s} \alpha_r \alpha_s \left[\sum_{l} \alpha_l \|\mathbf{a}_r - \mathbf{a}_s\|^2 \right] \\ &= \sum_{r<s} \alpha_r \alpha_s \|\mathbf{a}_r - \mathbf{a}_s\|^2 \sum_{l} \alpha_l \\ &= \sum_{r<s} \alpha_r \alpha_s \|\mathbf{a}_r - \mathbf{a}_s\|^2 = T_3. \end{aligned}$$

Una vez demostrada la igualdad (4.5), es evidente que $W(\mathbf{z}_0) \geq 0$ quedando demostrado el punto 1 de la proposición.

El punto 2 de la proposición supone la situación $k > q$.

Por una parte, $\psi(\mathbf{z}_0) \in \mathbb{R}^q$ y $\mathbf{a} = \mathbf{a}_k + \sum_{r=1}^{k-1} \alpha_r(\mathbf{a}_r - \mathbf{a}_k)$.

Por otra parte, $\mathbf{a}_1 - \mathbf{a}_k, \ldots, \mathbf{a}_{k-1} - \mathbf{a}_k$ forman una base de $\mathbb{R}^{k-1}$ (suponemos que son linealmente independientes sin falta de generalidad).

Como $q < k$, $\psi(\mathbf{z}_0) \in \mathbb{R}^{k-1}$ y fijado un origen en $\mathbb{R}^{k-1}$, por ejemplo, $\mathbf{a}_k$,

$$\exists \beta_r \in \mathbb{R}\,(r = 1, \ldots, k-1) \,:\, \psi(\mathbf{z}_0) = \mathbf{a}_k + \sum_{r=1}^{k-1} \beta_r(\mathbf{a}_r - \mathbf{a}_k).$$

Por tanto,

$$\begin{aligned} W(\mathbf{z}_0) &= \min_{\sum \alpha_r = 1} \|\psi(\mathbf{z}_0) - \mathbf{a}\|^2 \\ &= \min_{\alpha_1, \ldots, \alpha_{k-1}} \|\mathbf{a}_k + \sum_{r=1}^{k-1} \beta_r(\mathbf{a}_r - \mathbf{a}_k) - (\mathbf{a}_k + \sum_{r=1}^{k-1} \alpha_r(\mathbf{a}_r - \mathbf{a}_k))\|^2 = 0. \end{aligned}$$

Finalmente, si $\psi(\mathbf{z}_0)$ es colineal con $\mathbf{a}_1, \ldots, \mathbf{a}_k$ como se indica en el punto 3 de la proposición, otra vez, $\exists \beta_r \in \mathbb{R}\,(r = 1, \ldots, k-1) \,:\, \psi(\mathbf{z}_0) = \mathbf{a}_k + \sum_{r=1}^{k-1} \beta_r(\mathbf{a}_r - \mathbf{a}_k)$, concluyendo que $W(\mathbf{z}_0) = 0$. $\square$

El estadístico INCA coincide en algunos casos con estadísticos ya conocidos, tal como se presenta en las siguientes proposiciones.

Proposición 4.3.

Para $k = 2$, el estadístico $W(\mathbf{z}_0)$ se reduce al estadístico propuesto en Cuadras and Fortiana (2000).

Demostración:

En este caso $M = 2\Delta_{12}^2$ y $N = \Delta_{12}^2 + \phi_2^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0)$, de manera que

$$\alpha_1 = \frac{\Delta_{12}^2+\phi_2^2(\mathbf{z}_0)-\phi_1^2(\mathbf{z}_0)}{2\Delta_{12}^2} \quad \text{y} \quad \alpha_2 = 1 - \alpha_1.$$

Por tanto,

$$\begin{aligned} W(\mathbf{z}_0) &= \alpha_1\phi_1^2(\mathbf{z}_0) + \alpha_2\phi_2^2(\mathbf{z}_0) - \alpha_1\alpha_2\Delta_{12}^2 \\ &= \phi_2^2(\mathbf{z}_0) - \alpha_1(\Delta_{12}^2 + \phi_2^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0)) + \alpha_1^2\Delta_{12}^2 \\ &= \phi_2^2(\mathbf{z}_0) - \frac{\Delta_{12}^2 + \phi_2^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0)}{2\Delta_{12}^2}(\Delta_{12}^2 + \phi_2^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0)) \\ &+ \left(\frac{\Delta_{12}^2 + \phi_2^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0)}{2\Delta_{12}^2}\right)^2 \Delta_{12}^2 \\ &= \phi_2^2(\mathbf{z}_0) - \frac{\Delta_{12}^2 + \phi_2^2(\mathbf{z}_0) - \phi_1^2(\mathbf{z}_0)}{4\Delta_{12}^2}. \quad \square \end{aligned}$$

Proposición 4.4.

Sea $\mathbf{Z}_r \sim N(\mu_r, \Sigma_r)$ un vector p-dimensional que representa al cluster C_r, $r = 1, ..., k$. Consideremos el caso homocedástico $\Sigma_r = \Sigma$, $r = 1, ..., k$, y supongamos también que la distancia definida sobre los objetos es la distancia de Mahalanobis. Entonces, $W(\mathbf{z}_0)$ se reduce al estadístico propuesto por Rao (1962).

Demostración:

En la proposición 4.2 ya hemos visto que

$$L(\mathbf{z}_0) = \|\psi(\mathbf{z}_0) - \sum_{r=1}^{k} \alpha_r E(\psi(\mathbf{Z}_r))\|^2 , \quad \text{con} \quad \sum_{r=1}^{k} \alpha_r = 1.$$

Por otra parte, en esta situación, la función que proporciona las coordenadas principales es $\psi(\mathbf{z}_0) = \Sigma^{-1/2}\mathbf{z}_0$. Luego,

$$\begin{aligned} L(\mathbf{z}_0) &= \|\Sigma^{-1/2}(\mathbf{z}_0 - \sum_{r=1}^{k} \alpha_r \mu_r)\|^2 \\ &= (\mathbf{z}_0 - \sum_{r=1}^{k} \alpha_r \mu_r)' \Sigma^{-1} (\mathbf{z}_0 - \sum_{r=1}^{k} \alpha_r \mu_r), \end{aligned}$$

y por tanto

$$W(\mathbf{z}_0) = \min_{\sum \alpha_r = 1} L(\mathbf{z}_0) = \min_{\sum \alpha_r = 1} (\mathbf{z}_0 - \sum_{r=1}^{k} \alpha_r \mu_r)' \Sigma^{-1} (\mathbf{z}_0 - \sum_{r=1}^{k} \alpha_r \mu_r) . \quad \square$$

4.2. Estimación del número de clusters

Para abordar el problema de la estimación del número de clusters consideremos que los n objetos están clasificados en k clusters $C_1, \ldots, C_k$ de tamaños $n_1, \ldots, n_k$, respectivamente. Fijamos un cluster C_r y para cada objeto $\mathbf{z}$ del conjunto de datos definimos $W_{C_r}(\mathbf{z})$ como el valor del estadístico (4.1) calculado sobre el objeto $\mathbf{z}$ y respecto al nuevo cluster de δ-media $\sum_{s \neq r} \alpha_s E(\psi(\mathbf{Z}_s))$, donde

$\mathbf{Z}_s$ representa al cluster C_s $(s \neq r)$. Sea W_{C_r} el máximo de estos valores para todos los objetos que no están en C_r, es decir, $W_{C_r} = \text{máx}_{\mathbf{z} \notin C_r} W_{C_r}(\mathbf{z})$. Entonces, consideramos el siguiente criterio:

Criterio:

El objeto $\mathbf{z}$ de C_r está bien clasificado en C_r si $W_{C_r}(\mathbf{z}) > W_{C_r}$. El objeto $\mathbf{z}$ de C_r está mal clasificado en C_r si $W_{C_r}(\mathbf{z}) \leq W_{C_r}$, ya que esta desigualdad está indicando que el objeto está más cerca de algún otro cluster.

Sea N_r el total de objetos de C_r que están bien clasificados. Se define el índice $INCA_k$, asociado a la partición $C_1, ..., C_k$ como la probabilidad de los objetos bien clasificados:

$$INCA_k = \frac{1}{k} \sum_{r=1}^{k} \frac{N_r}{n_r}.$$

Valores de $INCA_k$ cercanos a 1, indican que la mayoría de los objetos están bien clasificados, y por tanto el valor de k refleja el número correcto de clusters. Por otra parte, valores de $INCA_k$ cercanos a 0, indican que la mayoría de los objetos están mal clasificados y por tanto el valor de k no se corresponderá con el del número adecuado de clusters. Por ello al representar los valores de $INCA_k$ contra k para $k = 2, \ldots, K$, con K fijado por el investigador, el número de clusters ideal se selecciona como aquel valor de k en el que el gráfico muestra el mayor salto decreciente. Por tanto, después del valor ideal de k, $INCA_k$ decrece rápidamente

y continúa decreciendo a medida que los clusters se van separando en clusters de menor tamaño. Es conocido que los resultados obtenidos por un procedimiento automatizado no deben ser aceptados sin ser previamente examinados y comprobar que realmente tienen sentido. Cuando el número de objetos es grande parece razonable aceptar máximos locales y tenerlos en consideración a la hora de decidir el número ideal de clusters, examinando por lo tanto las posibles particiones de los objetos con mayor detalle. Por todo ello, para conjuntos grandes de objetos a clasificar, se propone analizar todos los valores de k que corresponden a grandes saltos del valor de $INCA_k$. Hay que hacer notar que gráficas de valores bajos de $INCA_k$ y prácticamente planas, indican que no existe estructura de clusters o que todos los objetos pertenecen a un mismo cluster. En el caso de que los clusters sean co-lineales o co-planos, aunque ésta no sea una situación demasiado importante desde el punto de vista biomédico, o bien en el caso de que el espacio de los objetos tenga dimensión pequeña, se propone el siguiente método de partición de los objetos, buscando en cada paso únicamente dos clusters. Partiendo de todos los objetos, éstos se separan por algún procedimiento de partición en dos clusters C_1 y C_2, y se evalúa el valor de $INCA_2$. Después se vuelven a dividir los clusters C_1 y C_2 en dos nuevos clusters $C_{1(1)}$, $C_{2(1)}$ y $C_{1(2)}$, $C_{2(2)}$ respectivamente. Se buscan los valores asociados de $INCA_2$. Se repite dicho proceso hasta que los valores de $INCA_2$ presentan el mayor salto decreciente. De todas maneras, está previsto un mayor estudio al respecto ya que podrían valorarse diferentes combinaciones de pares de clusters para calcular $INCA_2$.

En resumen, al representar los valores de $INCA_k$ contra k para $k = 2, \ldots, K$,

con K fijado por el investigador, el número de clusters ideal se selecciona como aquel valor de k en el que el gráfico muestra el mayor salto decreciente. Cuando los valores de $INCA_k$ $(k = 2, \ldots, K)$ son bajos y sin pronunciados altibajos se concluye que los datos no presentan una estructura de cluster o bien que todos los objetos forman un único cluster.

4.3. Problema de la tipicalidad

Supongamos que después de un análisis de cluster ya está determinado el número de clusters de los datos. Sea $\mathbf{z}_0$ una nueva observación, ahora interesa decidir si $\mathbf{z}_0$ pertenece a alguno de los clusters C_r, $r = 1, ..., k$ ya fijados o, si por el constrario, $\mathbf{z}_0$ es un objeto outlier o atípico que pertenece a algún otro cluster desconocido. Sea pues el test de hipótesis,

$$H_0 \quad : \quad \mathbf{z}_0 \text{ proviene del cluster con } \delta\text{-media } \sum_{r=1}^{k} \alpha_r E(\psi(\mathbf{Z}_r)), \quad \sum_{r=1}^{k} \alpha_r = 1,\ r = 1, ..., k,$$

$$H_1 \quad : \quad \mathbf{z}_0 \text{ proviene de un cluster desconocido,}$$

y calculemos el estadístico dado en (4.1). Un valor significativo de $W(\mathbf{z}_0)$ indica que $\mathbf{z}_0$ proviene de un cluster diferente y desconocido. Si no es siginificativo, se clasifica el objeto $\mathbf{z}_0$ en C_r según el siguiente criterio:

Criterio:

$$\text{Se clasifica } \mathbf{z}_0 \text{ en } C_r \text{ si } U_r(\mathbf{z}_0) = \min_{s=1,\ldots,k} \{U_s(\mathbf{z}_0)\}, \tag{4.6}$$

donde $U_s(\mathbf{z}_0) = \phi_s^2(\mathbf{z}_0) - W(\mathbf{z}_0)$, $s = 1, ..., k$.

Según la proposición 4.1, $U_r(\mathbf{z}_0)$ representa la proyección (al cuadrado) de $\{\mathbf{z}_0, \mathbf{a}_r\}$ sobre el hiperplano $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$, con $\mathbf{a}_r$ la δ-media del cluster C_r, $r = 1, \ldots, k$. Ver la Figura 4.1, donde por simplicidad se ha denotado la proyección (al cuadrado) $U_r(\mathbf{z}_0)$ como p_r, $r = 1, ..., k$. Por tanto, el criterio (4.6) se corresponde con el siguiente criterio de clasificación más geométrico e intuitivo:

Clasificar $\mathbf{z}_0$ en C_r si la proyeccion $p_r = U_r(\mathbf{z}_0)$ es la menor de todas.

Cabe destacar que aunque las distribuciones muestrales de $W(\mathbf{z}_0)$ y $U_r(\mathbf{z}_0)$ $(r = 1, ..., k)$ resultan muy dificiles de encontrar para datos mixtos, éstas se pueden obtener mediante técnicas de remuestreo, en particular obteniendo muestras bootstrap como sigue. Se seleccionan N objetos $\mathbf{z}$ con reemplazamiento de la unión de $C_1, \ldots, C_k$ y se calculan los correspondientes valores $W(\mathbf{z})$ y $U_r(\mathbf{z})$ $(r = 1, ..., k)$. Se repite este proceso $10P$ veces. De esta manera se obtienen distribuciones bootstrap bajo H_0.

4.4. Estudio de simulación

Para poder evaluar la bondad de los métodos propuestos en este capítulo se ha llevado a cabo un estudio de simulación bajo ciertas circunstancias relevantes.

4.4.1. Resultados del estudio de simulación para la estimación del número de clusters

Con el fin de poder evaluar la eficacia del método propuesto para la estimación del número de clusters hemos generado diferentes conjuntos de datos a partir de una variedad de modelos incluyendo datos mixtos y diferentes números de variables. También hemos incluido dos modelos (modelos 4 y 5) con algunas variables irrelevantes o de ruido con el fin de oscurecer la estructura de clusters subyacente. Para los datos mixtos se ha utilizado la única función distancia que se puede aplicar con datos ausentes, es decir, la distancia de Gower y para las frecuencias la distancia de Bhattacharyya (ver capítulo 2). Para cada modelo se han generado cincuenta conjuntos de datos.

Modelo 1 Un único cluster en dimensión $p = 10$. A partir de la distribución multinomial $M(50, \pi_l = 1/10, l = 1, ..., 10)$, se han simulado $n = 200$ objetos. Cada objeto i se caracteriza por un vector p-dimensional $(f_{i1}, ..., f_{ip})$ cuyas coordenadas son las frecuencias generadas de manera que

$$f_{il} \geq 0,\ l = 1, ..., p \ \text{ y } \ \sum_{l=1}^{p} f_{il} = 1; \quad i = 1, ..., n.$$

Definimos la distancia entre dos objetos $i = (f_{i1}, ..., f_{ip})$ y $j = (f_{j1}, ..., f_{jp})$ como la distancia de Bhattacharyya (ver (2.1) del capítulo 2).

Modelo 2 Tres clusters en dimensión $p = 10$. Cada cluster se ha generado como frecuencias obtenidas a partir de una distribución multinomial. Cluster $C_1 \sim M(50, \pi_l = 1/10, l = 1, ..., 10)$; Cluster $C_2 \sim M(50, \pi_1 = ... = \pi_5 =$

$1/50, \pi_6 = ... = \pi_{10} = 9/50$); Cluster $C_3 \sim M(50, \pi_1 = \pi_2 = 9/20, \pi_3 = ... = \pi_{10} = 1/80)$. El tamaño de cada cluster se ha elegido al azar entre tamaños de 25 ó 50 objetos. Igual que en el caso anterior, se ha elegido la distancia de Bhattacharyya como distancia entre pares de objetos.

Modelo 3 Cuatro clusters en $p = 8$ dimensiones. Los valores correspondientes a cada objeto se han obtenido a partir de una variable continua, cuatro binarias y tres categóricas. La variable continua X sigue una distribución normal de manera que su media ha sido elegida al azar utilizando una distribución $N(0, 5^2)$. Las variables categóricas tienen 3, 4 y 5 categorías, respectivamente, y la descripción completa de los valores de los parámetros para las distribuciones que generan los datos de cada cluster se muestran en la Tabla 4.1. El tamaño de cada cluster se ha elegido al azar tomando 25 ó 50 objetos.

En esta situación cada objeto está caracterizado por $p_1 = 1$ variable continua, $p_2 = 4$ variables binarias y $p_3 = 3$ variables cualitativas y dado que los datos son mixtos, se ha tomado como distancia la denominada distancia de Gower (ver (2.3) del capítulo 2).

Modelo 4 Cuatro clusters en dimensión $p = 10$ con dos variables de ruido. Similar al modelo 3 pero se han añadido 2 variables de ruido simuladas a partir de la distribución $N(0, 1)$. Tal como en el modelo 3, se ha considerado la distancia de Gower como la distancia entre dos objetos.

Modelo 5 Cuatro clusters en dimensión $p = 13$ con variables de ruido. Similar al modelo 3 pero se han añadido 5 variables de ruido simuladas a partir

Tabla 4.1: Valores de los parámetros para los clusters C_r, $r = 1, ..., 4$ del modelo 3. En cada caso, la media μ_r ha sido seleccionada al azar utilizando una distribución $N(0, 5^2)$ $(r = 1, ..., 4)$.

Cluster C_1	Cluster C_2
$X \sim N(\mu_1, 1)$	$X \sim N(\mu_2, 1)$
$B_1 \sim B(27/30)$	$B_1 \sim B(1/30)$
$B_2 \sim B(1/30)$	$B_2 \sim B(27/30)$
$B_3 \sim B(1/30)$	$B_3 \sim B(1/30)$
$B_4 \sim B(1/30)$	$B_4 \sim B(1/30)$
$Q_1 \sim (90/100, 5/100, 5/100)$	$Q_1 \sim (5/100, 90/100, 5/100)$
$Q_2 \sim (27/30, 1/30, 1/30, 1/30)$	$Q_2 \sim (1/30, 27/30, 1/30, 1/30)$
$Q_3 \sim (12/30, 15/30, 1/30, 1/30, 1/30)$	$Q_3 \sim (1/30, 12/30, 15/30, 1/30, 1/30)$

Cluster C_3	Cluster C_4
$X \sim N(\mu_3, 1)$	$X \sim N(\mu_4, 1)$
$B_1 \sim B(1/30)$	$B_1 \sim B(1/30)$
$B_2 \sim B(1/30)$	$B_2 \sim B(1/30)$
$B_3 \sim B(27/30)$	$B_3 \sim B(1/30)$
$B_4 \sim B(1/30)$	$B_4 \sim B(27/30)$
$Q_1 \sim (5/100, 5/100, 90/100)$	$Q_1 \sim (1/30, 1/30, 1/30)$
$Q_2 \sim (1/30, 1/30, 27/30, 1/30)$	$Q_2 \sim (1/30, 1/30, 1/30, 27/30)$
$Q_3 \sim (1/30, 1/30, 12/30, 15/30, 1/30)$	$Q_3 \sim (1/30, 1/30, 1/30, 12/30, 15/30)$

de la distribución $N(0, 1)$. Tal como en el modelo 3, se ha considerado la distancia de Gower como la distancia entre dos objetos.

La bondad del indice INCA en este estudio de simulación se ha comparado con la obtenida con el índice silhouette. Este índice fue propuesto por Rousseeuw (1987) y se resume de la siguiente manera. Para cada par de objetos i y j, la distancia entre ellos está definida por $\delta(i,j)$. Por una parte se considera la distancia media del i-ésimo objeto del cluster A al resto de objetos del mismo cluster $a_i = \sum_{j\in A, j\neq i} \delta(i,j)/(n_A - 1)$ donde n_A es el número de objetos en A. Por otra parte, se considera la distancia media del objeto i a los objetos de cada uno de los clusters (designado por C) diferentes de A, $d(i,C) = \sum_{j\in C} \delta(i,j)/n_C$. La distancia media a los objetos del cluster más cercano se denota por b_i, con $b_i = \text{mín}_{C\neq A}\{d(i,C)\}$. Por último, el valor silhouette del objeto i se define como $s(i) = (b_i - a_i)/\text{máx}\{a_i, b_i\}$. A la hora se estimar el número de clusters, Rousseeuw (1987) propone seleccionar aquel número k que tenga asociada la partición con el mayor valor medio del silhouette $\overline{s} = \sum_i s(i)/n$. Hemos escogido esta medida para las comparaciones con el método que se propone en este capítulo de la memoria porque es el único, entre todos los revisados en Dudoit and Fridlyand (2002), que puede trabajar con cualquier tipo de datos. A fin de construir una partición inicial, hemos considerado el método de cluster Partitioning Around Medoids (PAM) (Kaufman and Rousseeuw, 1990) para obtener particiones de los datos en 2, 3, ..., 9 y 10 clusters. Hemos seleccionado este método de cluster porque a diferencia de otros métodos particionales, éste necesita únicamente las distancias entre los objetos para su aplicación. Los resultados obtenidos se muestran en la Tabla 4.2.

Tabla 4.2: Distribución del número de clusters estimado por el método silhouette y el método INCA, respectivamente, para cada uno de los cinco modelos simulados. El verdadero número de clusters se ha señalado con un asterisco.

Procedimiento	Número k de clusters					
Modelo 1	1*	2	3	4	5	> 5
Silhouette	-	-	-	-	-	-
$INCA_k$	50	0	0	0	0	0
Modelo 2	1	2	3*	4	5	> 5
Silhouette	-	50	0	0	0	0
$INCA_k$	0	0	49	1	0	0
Modelo 3	1	2	3	4*	5	> 5
Silhouette	-	0	0	50	0	0
$INCA_k$	0	0	0	48	2	0
Modelo 4	1	2	3	4*	5	> 5
Silhouette	-	0	0	50	0	0
$INCA_k$	0	0	0	49	1	0
Modelo 5	1	2	3	4*	5	> 5
Silhouette	-	0	0	50	0	0
$INCA_k$	0	0	0	50	0	0

Se puede ver que INCA ofrece en general buenos resultados para todos los modelos considerados. Para el modelo 1 no es posible hacer comparaciones ya

que el índice silhouette no puede estimar situaciones con un único cluster. Para el modelo 2, el índice silhouette no obtiene buenos resultados mientras que INCA sólo ha fallado en una simulación. Cabe destacar que el estadístico silhouette compara la distancia media de cada objeto a objetos de su cluster con la distancia media a objetos del cluster más cercano. Esto, en esta situación particular donde el cluster C_3 está alejado de los clusters C_1 y C_2, siendo éstos dos los más próximos, explica los malos resultados del índice silhouette. Es decir, al considerar la partición en dos (partición $\{C_1 \cup C_2, C_3\}$), el índice silhouette es alto porque los dos clusters de la partición están lejos. Sin embargo, al considerar la partición en tres (partición $\{C_1, C_2, C_3\}$), el índice silhouette no es tan alto porque los clusters C_1 y C_2 son vecinos próximos de manera los que valores silhouette $s(i)$ relativos a objetos $i \in C_1, C_2$ son más pequeños y como consecuencia, también el valor silhouette medio $\overline{s}$.

Para los modelos 3 y 4, INCA y silhouette ofrecen resultados similares aunque INCA ha fallado en una y dos simulaciones, respectivamente. Por último, para el modelo 5 los dos procedimientos han estimado correctamente el número de clusters en las 50 simulaciones.

4.4.2. Resultados del estudio de simulación para el caso de la tipicalidad

A continación se considera el problema de identificar un objeto atípico. En este caso es importante tener una idea de cómo es el error de Tipo I y la potencia

del test INCA. Con este objetivo, se ha aplicado el test INCA a diferentes tipos de distribuciones y diferentes tamaños muestrales. El error de tipo I mide en qué medida rechazamos la hipótesis nula cuando en realidad los datos cumplen esta hipótesis nula. Por otro lado, el test debería tener la capacidad de rechazar la hipótesis nula cuando ésta es falsa. Se denomina potencia del test a la medida con la que el test rechaza la hipótesis nula en estas circustancias. Así pues, las simulaciones para medir el error de tipo I sólo involucran a clusters con δ-media igual a una combinación lineal de las δ-medias de los clusters prefijados C_r, $r = 1, \ldots, k$. En las simulaciones para medir la potencia, la hipótesis alternativa debe ser cierta por lo que el nuevo cluster debe tener una δ-media que no sea combinación lineal de las δ-medias de C_r, $r = 1, \ldots, k$.

Consideremos pues los modelos 2, 3, 4 y 5 detallados en el apartado anterior. Como los parámetros involucrados en el estadístico INCA dado en (4.1) son desconocidos, éstos se han estimado a partir de los datos y puesto que la estabilidad de las estimaciones de los parámetros depende del tamaño muestral (McLachlan, 1982), se han simulado clusters de diferentes tamaños.

Para evaluar el error de tipo I del test, hemos generado para cada uno de los citados modelos 1000 objetos nuevos de cada cluster $C_1, \ldots, C_k$ y se les ha aplicado el test INCA. Es decir, para cada objeto se ha decidido si éste provenía de un cluster con δ-media $\sum_{r=1}^{k} \alpha_r E(\psi(\mathbf{Z}_r)$ o no, donde $E(\psi(\mathbf{Z}_r)$ es la δ-media del cluster C_r $(r = 1, ..., k)$ y $\sum_{r=1}^{k} \alpha_r = 1$. En caso que se decida que el objeto proviene de un cluster con la citada δ-media, entonces lo hemos clasificado en uno de los clusters $C_1, \ldots, C_k$ según el criterio (4.6). A continuación hemos com-

binado los resultados del porcentaje de clasificación correcto en cada cluster para obtener el porcentaje global de clasificación de los objetos en el cluster correcto. Cuando no es posible ninguna asignación a un cluster, el objeto es considerado simplemente como atípico.

Para evaluar la potencia del test, de forma análoga se han generado 1000 objetos nuevos de cada cluster C_r $(r = 1, ..., k)$ y se les ha aplicado el test INCA para decidir si provenían del cluster con δ-media $\sum_{s \neq r} \alpha_s E(\psi(\mathbf{Z}_s)$ donde $\sum_{s \neq r} \alpha_s = 1$.

A continuación se pasan a comentar los resultados obtenidos tomando siempre como nivel de significación $\alpha = 0.05$. Los resultados obtenidos sobre el error de tipo I se muestran en las Tablas 4.3, 4.4, 4.5 y 4.6. Las dos últimas columnas de dichas Tablas indican el nivel (en porcentaje) del test INCA para cada cluster y el nivel global considerando todos los clusters respectivamente. Se observa que excepto para el modelo 2, el nivel del test para cada cluster está entre 5 % y 10 % en el caso de muestras pequeñas y alrededor de 5 % o menor en el caso de muestras grandes. El nivel global obtenido conjuntamente para todos los clusters está entre 5 y 11 por ciento para muestras pequeñas y entorno a 5 por ciento o menor, para muestras grandes. Obsérvese que para el modelo 3 el nivel del test decrece al aumentar el tamaño muestral del cluster C_4. Como en este cluster la variable Q_1 tiene el mismo peso para todas las categorías, la variabilidad del cluster C_4 es considerable por lo que la mejoría que se observa en el nivel del test se debe al aumento de observaciones de esta variable Q_1. Para el modelo 2 y cluster C_1, el nivel del test no está controlado y los niveles individuales están entre 20 y 30 por ciento, siendo menores para tamaños muestrales grandes. Como en C_1

todas las dimensiones tienen la misma importancia mientras que en el cluster C_3 sólo las dos primeras son importantes, era esperable un nivel alto para tamaños muestrales pequeños (tamaño 10). En este modelo, el nivel global está entre 7 y 15 por ciento, siendo menor para tamaños muestrales grandes.

Una vez decidido qué objetos son típicos, es decir, que pertenecen a alguno de los grupos iniciales, los objetos deben ser clasificados en uno de ellos siguiendo el criterio (4.6). Las Tablas 4.7, 4.8, 4.9 y 4.10 muestran buenos resultados respecto a esta clasificación. Así se observa que el porcentaje global de buena clasificación es en todos los modelos superior al 85 %, siendo ligeramente superior al aumentar el tamaño de las muestras. Los resultados relativos al estudio de la potencia del test se muestran en las Tablas 4.11, 4.12, 4.13 y 4.14. Como indica la última columna de estas Tablas, los resultados obtenidos son muy buenos, tomando la potencia del test valores entre el 93 % y el 100 %. Todos estos resultados muestran la eficacia y buen funcionamiento de los procedimientos desarrollados.

Tabla 4.3: Estudio del error de tipo I del test INCA utilizando datos generados usando el modelo 2 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. Las dos últimas columnas muestran, respecivamente, el nivel individual en cada grupo y el nivel global.

Tamaño muestral			Cluster real	Objetos típicos	Nivel del cluster (%)	Nivel global (%)
10	10	10	C_1	637	36.3	
			C_2	971	2.9	
			C_3	999	0.1	
						13.1
10	10	100	C_1	680	32.0	
			C_2	948	5.2	
			C_3	982	1.8	
						12.97
10	100	100	C_1	666	33.4	
			C_2	920	8.0	
			C_3	949	5.1	
						15.5
100	100	100	C_1	810	19.0	
			C_2	986	1.4	
			C_3	997	0.3	
						6.9

Tabla 4.4: Estudio del error de tipo I del test INCA utilizando datos generados usando el modelo 3 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. Las dos últimas columnas muestran, respecivamente, el nivel individual en cada grupo y el nivel global.

Tamaño muestral				Cluster real	Objetos típicos	Nivel del cluster (%)	Nivel global (%)
10	10	10	10	C_1	871	12.9	
				C_2	895	10.5	
				C_3	889	11.1	
				C_4	895	10.5	
							11.25
10	10	10	100	C_1	974	2.6	
				C_2	988	1.2	
				C_3	976	2.4	
				C_4	975	2.5	
							2.18
10	10	100	100	C_1	948	5.2	
				C_2	953	4.7	
				C_3	955	4.5	
				C_4	943	5.7	
							5.03
10	100	100	100	C_1	959	4.1	
				C_2	975	2.5	
				C_3	971	2.9	
				C_4	965	3.5	
							3.25
100	100	100	100	C_1	969	3.1	
				C_2	974	2.6	
				C_3	981	1.9	
				C_4	965	3.5	
							2.78

Tabla 4.5: Estudio del error de tipo I del test INCA utilizando datos generados usando el modelo 4 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. Las dos últimas columnas muestran, respecivamente, el nivel individual en cada grupo y el nivel global.

Tamaño muestral				Cluster real	Objetos típicos	Nivel del cluster (%)	Nivel global (%)
10	10	10	10	C_1	970	3.0	
				C_2	943	5.7	
				C_3	964	3.6	
				C_4	934	6.6	
							4.72
10	10	10	100	C_1	932	6.8	
				C_2	944	5.6	
				C_3	951	4.9	
				C_4	939	6.1	
							5.85
10	10	100	100	C_1	969	3.1	
				C_2	943	5.7	
				C_3	979	2.1	
				C_4	960	4.0	
							3.72
10	100	100	100	C_1	929	7.1	
				C_2	929	7.1	
				C_3	951	4.9	
				C_4	900	10.0	
							7.27
100	100	100	100	C_1	955	4.5	
				C_2	948	5.2	
				C_3	967	3.3	
				C_4	938	6.2	
							4.8

Tabla 4.6: Estudio del error de tipo I del test INCA utilizando datos generados usando el modelo 5 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. Las dos últimas columnas muestran, respecivamente, el nivel individual en cada grupo y el nivel global.

Tamaño muestral				Cluster real	Objetos típicos	Nivel del cluster (%)	Nivel global (%)
10	10	10	10	C_1	962	3.8	
				C_2	983	1.7	
				C_3	977	2.3	
				C_4	974	2.6	
							2.6
10	10	10	100	C_1	944	5.6	
				C_2	967	3.3	
				C_3	956	4.4	
				C_4	960	4.0	
							4.32
10	10	100	100	C_1	953	4.7	
				C_2	956	4.4	
				C_3	978	2.2	
				C_4	947	5.3	
							4.15
10	100	100	100	C_1	947	5.3	
				C_2	970	3.0	
				C_3	966	3.4	
				C_4	959	4.1	
							3.95
100	100	100	100	C_1	962	3.8	
				C_2	956	4.4	
				C_3	960	4.0	
				C_4	936	6.4	
							4.65

Tabla 4.7: Clasificación de los objetos identificados como típicos por el test INCA utilizando datos generados usando el modelo 2. También se indica el número de objetos que han sido identificados como atípicos. Para cada grupo se han generado 1000 objetos. Las dos últimas columnas muestran respectivamente el porcentaje de buena clasificacioń individual para los objetos de cada grupo y para todos los objetos conjuntamente.

Tamaño muestral			Cluster real	Asignado al cluster				Porc. clas. por clusters	Porc. clas. global
				C_1	C_2	C_3	Atípico		
10	10	10	C_1	637	0	0	363	63.7	
			C_2	0	971	0	29	97.1	
			C_3	0	0	999	1	99.9	
									86.9
10	10	100	C_1	680	0	0	319	68.0	
			C_2	2	946	0	52	94.6	
			C_3	0	0	982	18	98.2	
									86.9
10	100	100	C_1	666	0	0	384	66.6	
			C_2	0	920	0	80	92.0	
			C_3	0	0	949	51	94.9	
									84.5
100	100	100	C_1	810	0	0	190	81.0	
			C_2	1	985	0	14	98.5	
			C_3	0	0	997	3	99.7	
									93.1

Tabla 4.8: Clasificación de los objetos identificados como típicos por el test INCA utilizando datos generados usando el modelo 3. También se indica el número de objetos que han sido identificados como atípicos. Para cada grupo se han generado 1000 objetos. Las dos últimas columnas muestran respectivamente el porcentaje de buena clasificacioń individual para los objetos de cada grupo y para todos los objetos conjuntamente.

Tamaño muestral				Cluster real	Asignado al cluster					Porc. clas. por clusters	Porc. clas. global
					C_1	C_2	C_3	C_4	Atípico		
10	10	10	10	C_1	870	0	0	1	129	87.0	
				C_2	1	893	1	0	105	89.3	
				C_3	0	0	885	4	111	88.5	
				C_4	1	0	4	890	105	89.0	
											88.4
10	10	10	100	C_1	970	0	1	3	26	97.0	
				C_2	1	986	1	0	12	98.6	
				C_3	1	0	966	9	24	96.6	
				C_4	1	0	3	971	25	97.1	
											97.3
10	10	100	100	C_1	947	0	0	1	52	94.7	
				C_2	0	952	1	0	47	95.2	
				C_3	1	0	950	4	45	95.0	
				C_4	1	0	7	935	57	93.5	
											94.6
10	100	100	100	C_1	955	1	0	3	41	95.5	
				C_2	1	973	1	0	25	97.3	
				C_3	0	0	967	4	29	96.7	
				C_4	4	0	7	954	35	95.4	
											96.2
100	100	100	100	C_1	967	0	0	2	31	96.7	
				C_2	1	972	1	0	26	97.2	
				C_3	2	0	974	5	19	97.4	
				C_4	3	0	7	955	35	95.5	
											96.7

Tabla 4.9: Clasificación de los objetos identificados como típicos por el test INCA utilizando datos generados usando el modelo 4. También se indica el número de objetos que han sido identificados como atípicos. Para cada grupo se han generado 1000 objetos. Las dos últimas columnas muestran respectivamente el porcentaje de buena clasificacioń individual para los objetos de cada grupo y para todos los objetos conjuntamente.

Tamaño muestral				Cluster real	Asignado al cluster					Porc. clas. por clusters	Porc. clas. global
					C_1	C_2	C_3	C_4	Atípico		
10	10	10	10	C_1	960	3	5	2	30	96.0	
				C_2	0	941	0	2	57	94.1	
				C_3	0	1	963	0	36	96.3	
				C_4	3	0	18	913	66	91.3	
											94.4
10	10	10	100	C_1	928	1	1	2	68	92.8	
				C_2	2	940	0	2	56	94.0	
				C_3	1	0	947	3	49	94.7	
				C_4	4	0	5	930	61	93.0	
											93.6
10	10	100	100	C_1	962	1	4	2	31	96.2	
				C_2	0	941	0	2	57	94.1	
				C_3	0	2	974	3	21	97.4	
				C_4	3	0	6	951	40	95.1	
											95.7
10	100	100	100	C_1	926	0	1	2	71	92.6	
				C_2	0	928	0	1	71	92.8	
				C_3	0	0	948	3	49	94.8	
				C_4	4	0	5	891	100	89.1	
											92.3
100	100	100	100	C_1	952	0	1	2	45	95.2	
				C_2	1	946	0	1	52	94.6	
				C_3	0	1	963	3	33	96.3	
				C_4	4	0	5	929	62	92.9	
											94.7

Tabla 4.10: Clasificación de los objetos identificados como típicos por el test INCA utilizando datos generados usando el modelo 5. También se indica el número de objetos que han sido identificados como atípicos. Para cada grupo se han generado 1000 objetos. Las dos últimas columnas muestran respectivamente el porcentaje de buena clasificacioń individual para los objetos de cada grupo y para todos los objetos conjuntamente.

Tamaño muestral				Cluster real	Asignado al cluster					Porc. clas. por clusters	Porc. clas. global
					C_1	C_2	C_3	C_4	Atípico		
10	10	10	10	C_1	958	0	1	3	38	95.8	
				C_2	1	981	0	1	17	98.1	
				C_3	1	1	970	5	23	97.0	
				C_4	4	1	4	965	26	96.5	
											96.8
10	10	10	100	C_1	939	0	1	4	56	93.9	
				C_2	2	963	0	2	33	96.3	
				C_3	0	1	950	5	44	95.0	
				C_4	4	0	2	954	40	95.4	
											95.1
10	10	100	100	C_1	950	2	0	1	47	95.0	
				C_2	1	953	0	2	44	95.3	
				C_3	0	1	970	7	22	97.0	
				C_4	3	0	2	942	53	94.2	
											95.4
10	100	100	100	C_1	946	0	0	1	53	94.6	
				C_2	3	965	0	2	30	96.5	
				C_3	0	1	961	4	34	96.1	
				C_4	4	1	2	952	41	95.2	
											95.6
100	100	100	100	C_1	961	0	0	1	38	96.1	
				C_2	1	955	0	0	44	95.5	
				C_3	1	1	953	5	40	95.3	
				C_4	4	1	1	930	64	93.0	
											95.0

Tabla 4.11: Estudio de la potencia del test INCA utilizando datos generados usando el modelo 2 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. La última columna muestra valores de potencia superiores al 98.4 %.

Tamaño muestral		Clusters iniciales		Cluster testado	Objetos atípicos
10	10	C_1	C_2	C_3	1000
		C_1	C_3	C_2	1000
		C_2	C_3	C_1	984
10	100	C_1	C_2	C_3	1000
		C_1	C_3	C_2	1000
		C_2	C_3	C_1	1000
100	100	C_1	C_2	C_3	1000
		C_1	C_3	C_2	1000
		C_2	C_3	C_1	999

Tabla 4.12: Estudio de la potencia del test INCA utilizando datos generados usando el modelo 3 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. La última columna muestra valores de potencia superiores al 99.6 %.

Tamaño muestral			Clusters iniciales			Cluster testado	Objetos atípicos
10	10	10	C_1	C_2	C_3	C_4	1000
			C_1	C_2	C_4	C_3	1000
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	999
10	10	100	C_1	C_2	C_3	C_4	1000
			C_1	C_2	C_4	C_3	996
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	999
10	100	100	C_1	C_2	C_3	C_4	1000
			C_1	C_2	C_4	C_3	1000
			C_1	C_3	C_4	C_2	999
			C_2	C_3	C_4	C_1	1000
100	100	100	C_1	C_2	C_3	C_4	999
			C_1	C_2	C_4	C_3	999
			C_1	C_3	C_4	C_2	999
			C_2	C_3	C_4	C_1	1000

Tabla 4.13: Estudio de la potencia del test INCA utilizando datos generados usando el modelo 4 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. La última columna muestra valores de potencia superiores al 99.1 %.

Tamaño muestral			Clusters iniciales			Cluster testado	Objetos atípicos
10	10	10	C_1	C_2	C_3	C_4	992
			C_1	C_2	C_4	C_3	994
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	991
10	10	100	C_1	C_2	C_3	C_4	998
			C_1	C_2	C_4	C_3	996
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	999
10	100	100	C_1	C_2	C_3	C_4	992
			C_1	C_2	C_4	C_3	995
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	999
100	100	100	C_1	C_2	C_3	C_4	998
			C_1	C_2	C_4	C_3	997
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	1000

Tabla 4.14: Estudio de la potencia del test INCA utilizando datos generados usando el modelo 5 (al nivel $\alpha = 0.05$). Se han simulado 1000 objetos de cada grupo. La última columna muestra valores de potencia superiores al 93.4 %.

Tamaño muestral			Clusters iniciales			Cluster testado	Objetos atípicos
10	10	10	C_1	C_2	C_3	C_4	998
			C_1	C_2	C_4	C_3	993
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	983
10	10	100	C_1	C_2	C_3	C_4	998
			C_1	C_2	C_4	C_3	996
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	987
10	100	100	C_1	C_2	C_3	C_4	963
			C_1	C_2	C_4	C_3	995
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	996
100	100	100	C_1	C_2	C_3	C_4	934
			C_1	C_2	C_4	C_3	996
			C_1	C_3	C_4	C_2	1000
			C_2	C_3	C_4	C_1	1000

4.5. Aplicaciones a datos reales

Para poder valorar correctamente la eficacia de los procedimientos presentados sobre datos no-ideales, es necesario aplicarlos sobre conjuntos de datos reales que tengan una estructura conocida. A continuación se presentan los resultados obtenidos sobre dos estudios relativos a enfermedades dermatológicas y cáncer linfático, respectivamente.

4.5.1. Diagnóstico de enfermedades dermatológicas

Como ejemplo real para probar nuestro procedimiento hemos considerado la información accesible en las bases de datos de UCI Knowledge Discovery (Hettich and Bay, 1999), respecto a enfermedades dermatológicas que fue cedido por H. A. Guvenir (Dpt. Computer Engineering and Information Science, Bilkent University, Turkey). Esta información numérica ha sido muy utilizada en el contexto de la clasificación aplicando diferentes procedimientos, como por ejemplo Clark and Niblett (1987); Guvenir *et al.* (1998) o Luukka (2007). Sin embargo quedan importantes problemas médicos por responder. Así el diagnóstico diferencial de las enfermedades eritematoescamosas es un verdadero problema en Dermatología (Ferrandiz, 1996). El fichero de datos utilizado contiene casos de pacientes con diagnósticos o clases conocidas. Así, se reconocen en él diagnósticos diferentes compartiendo todos ellos algunas características clínicas, aunque con pequeñas diferencias. Las seis enfermedades que se incluyen en el fichero original son: Psoriasis, Dermatitis seborreica, Pitiriasis rosada, Liquen plano, Dermatitis crónica

y Pitiriasis rubra pilaris. Mientras que algunas características tienen mayor incidencia en ciertas enfermedades, en otras también pueden aparecer en algunos estadios de su desarrollo dificultando por tanto su diagnóstico. Normalmente es necesaria una biopsia para realizar dicho diagnóstico pero por desgracia estas enfermedades también comparten muchas características histopatológicas. Algunas de las características de estas patologías se discuten en Guvenir *et al.* (1998). El conjunto de datos contiene 366 casos u objetos presentando 34 características. De estas características 12 son clínicas (por ejemplo edad, historia familiar etc.), mientras que 22 son características histopatológicas obtenidas a partir de una biopsia de la piel del paciente. Respecto a las características de tipo clínico, la variable edad está expresada en años y presenta valores faltantes (o missing) en 8 casos. La historia familiar toma valores 1 o 0 dependiendo de si la enfermedad se ha observado o no en la familia del paciente. Otras variables clínicas o histopatológicas recogen una gradación en el rango 0-3, donde 0 indica ausencia de la correspondiente característica, 3 representa la mayor manifestación posible y los valores 1 y 2 indican niveles intermedios. Dado que estamos en presencia de datos mixtos y con valores faltantes, hemos utilizado la distancia ponderada de Gower, la cual, como ya se ha comentado anteriormente es adecuada para este tipo de situaciones (ver (2.4) del capítulo 2).

Con el fin de evaluar la bondad del índice INCA para estimar el número de clusters se ha procedido a obtener particiones de los 366 objetos en $2, \ldots, 9$ y 10 clusters mediante el procedimiento PAM. Como es habitual, la valoración de la partición obtenida se ha hecho mediante el cociente del número de casos

clasificados correctamente respecto al total de casos. En la Tabla 4.15 se muestra la partición obtenida para $k = 6$ clusters. De los 366 objetos sólo 25 (6.8 %) han sido mal clasificados.

Tabla 4.15: Partición obtenida sobre los datos de diagnóstico dermatológico con $k = 6$ y utilizando el método PAM. La partición obtenida se compara con la clasificación real en 6 grupos de los datos.

	Partición PAM ($k = 6$)						
Clase diagnóstico	1	2	3	4	5	6	Total
1- Psoriasis	105	7	0	0	0	0	112
2- Dermatitis seborreica	0	52	9	0	0	0	61
3- Pitiriasis rosada	0	5	44	0	0	0	49
4- Liquen plano	0	0	1	71	0	0	72
5- Dermatitis crónica	0	2	1	0	49	0	52
6- Pitiriasis rubra pilaris	0	0	0	0	0	20	20

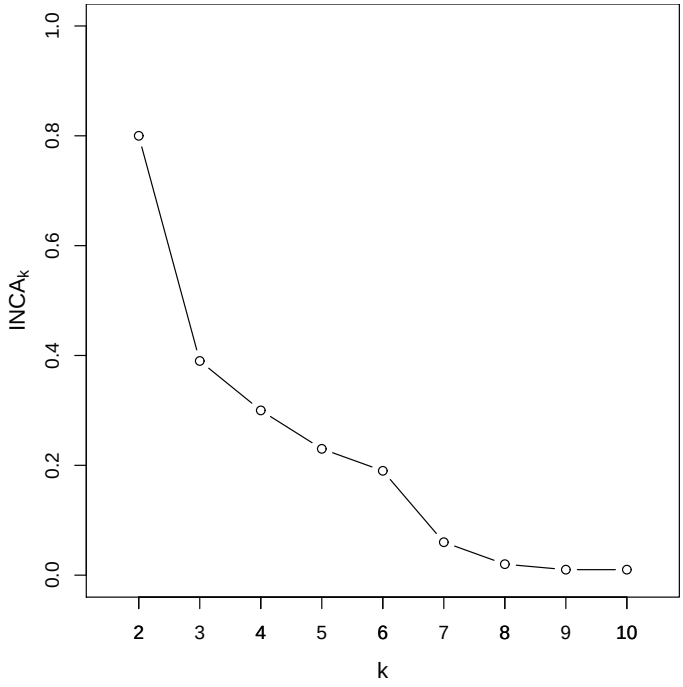


Figura 4.2: Estimación del número de clusters utilizando el índice INCA. Distribución de los valores $INCA_k$ frente a k aplicado a los datos de los diagnósticos dermatológicos. Se observan como candidatos para el número de clusters, en primera y segunda opción, los valores $k = 2$ y $k = 6$ respectivamente.

Los valores de $INCA_k$ frente a k ($k = 2, \ldots, 10$) se muestran en la Figura 4.2. Este gráfico muestra claramente que la mejor partición de los datos tiene dos clusters, ya que el mayor decrecimiento se produce al pasar de la partición para $k = 2$ a la partición para $k = 3$. Así se obtiene un cluster que contiene los 108 casos de Psoriasis y otro cluster con el resto de los casos. Esta primera partición en dos clusters es lógica desde un punto de vista médico, puesto que la Psoriasis se diferencia claramente de las demás patologías estudiadas (Griffiths and Barker, 2007). Como el conjunto de datos es grande (366 objetos) también se han mirado posibles valores de k para los cuales el índice $INCA_k$ presenta un decrecimiento notable. En este caso, el índice $INCA_k$ muestra un salto importante para $k = 6$. Por tanto, el valor $k = 6$ debe ser también considerado como un valor óptimo del número de clusters. El gráfico de $INCA_k$ versus k parece indicar ciertas particiones jerárquicas. Primero se separa la Psoriasis, separándose el resto de grupos para $k = 6$. Como los datos son mixtos hemos comparado estos resultados con los obtenidos utilizando el índice silhouette. La Figura 4.3 muestra que el valor máximo de este índice se alcanza para $k = 3$, que corresponde a la partición de un cluster con los 108 casos de Psoariasis, otro cluster con los 71 casos de Linquen plano y un último con el resto de los casos. La partición para $k = 2$ es la segunda mejor partición que ofrece el índice silhouette y la partición en $k = 6$ se indica como la cuarta mejor partición. Puesto que en los datos originales hay 6 clusters, parece que el índice INCA ofrece mejores resultados para la estimación del número de clusters.

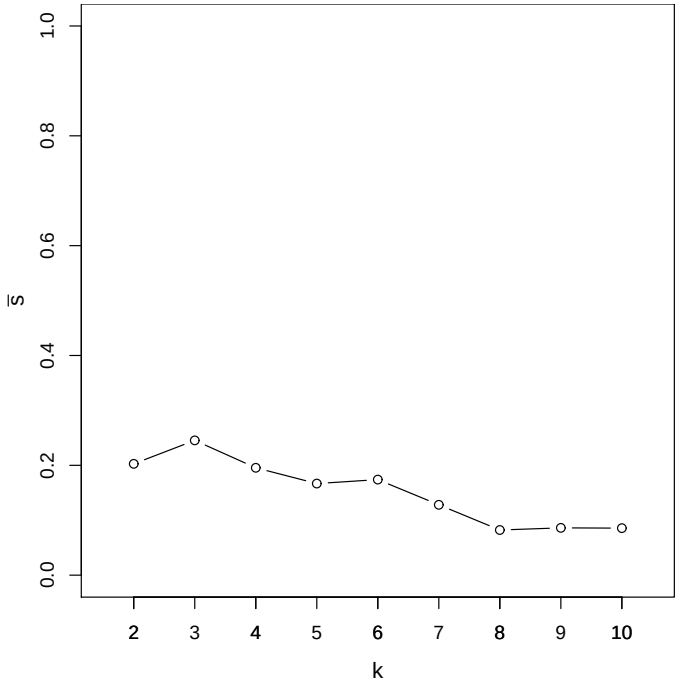


Figura 4.3: Estimación del número de clusters utilizando el índice silhouette. Distribución de los valores del índice frente a k aplicado a los datos de los diagnósticos dermatológicos. Se observan como candidatos para el número de clusters, en primera, segunda y cuarta opción, los valores $k = 3$, $k = 2$ y $k = 6$ respectivamente.

Por otra parte también hemos evaluado la bondad del test INCA utilizando este mismo conjunto de datos dermatológicos. Para ello se ha considerado uno de los grupos de datos, por ejemplo, Psoriasis, como casos de diagnóstico desconocido. A continuación se ha aplicado el test INCA para decidir si estos casos se consideran como típicos respecto de los grupos ya definidos (Dermatitis seborreica, Pitiriasis rosada, Liquen plano, Dermatitis crónica y Pitiriasis rubra pilaris), o bien se consideran como atípicos (tal como sabemos que son). Hemos repetido este mismo procedimiento cambiando cada vez el grupo de los casos considerados como desconocidos para todos los tipos de diagnóstico.

Los resultados recogidos en la Tabla 4.16 en general muestran buenos resultados al nivel 5 %. Cuando hemos considerado como desconocidos los casos de los clusters Psoriasis, Linquen plano o Pitiriasis rubra pilaris, el 100 % de los casos han sido detectados correctamente como atípicos. Al considerar como desconocidos los casos de Dermatitis crónica, 42 de ellos se han detectado como atípicos mientras que 10 como típicos. En cambio, para los casos de la Dermatitis seborreica o Pitiriasis rosada no se han obtenido buenos resultados. En esta ocasión, 56 de los 61 casos de Dermatitis seborreica se han detectado como típicos y han sido clasificados como casos de Pitiriasis rosada. De manera similar, todos los casos de Pitiriasis rosada se han detectado como típicos y han sido clasificados como casos de Dermatitis seborreica. Esto es debido a que ambas enfermedades son muy similares en cuanto a su sintomatología siendo muy difícil identificarlas como atípicas (Guvenir *et al.*, 1998). Cabe destacar que los dos grupos muestran para todos los casos el valor 0 en 6 variables. Solamente 1, 2 ó 3 casos de estos

grupos presentan un valor 1 en 9 variables tomando el valor 0 en el resto de los casos. Además, la variable edad no es significativamente diferente en los dos grupos.

Tabla 4.16: Estudio de la bondad del test INCA aplicado a los datos de diagnśtico dermatológico. Se ha considerado cada grupo previamente diagnosticado como grupo de diagnóstico desconocido y se ha evaluado cuántos casos son considerados correctamente como atípicos por el test INCA.

Cluster	Casos considerados	Casos detectados como atípicos
Psoriasis	112	112
Dermatitis seborreica	61	5
Liquen plano	72	72
Pitiriasis rosada	49	0
Dermatitis crónica	52	42
Pitiriasis rubra pilaris	20	20

Nota: Todos los casos son de un cluster atípico

Para acabar con este ejemplo supongamos fijados los 6 clusters de diagnósticos. Hemos seleccionado al azar 106 casos (33 Psoriasis, 25 Dermatitis seborreica, 25 Liquen plano, 10 Pitiriasis rosada, 8 Dermatitis crónica y 5 Pitiriasis rubra pilaris), dos de ellos con valores ausentes y se han considerado estos casos como de diagnóstico desconocido. Se les ha aplicado el test INCA para decidir si son típicos o atípicos, es decir, si estos casos provienen de un nuevo tipo de diagnóstico. Al nivel 5 % (ver Tabla 4.17) el test identifica como típicos casi todos los casos

(98 de los 106 casos, incluso los dos con valores faltantes) y sólo uno de ellos ha sido clasificado erróneamente. En esta situación, al estar presente los 6 tipos de diagnóstico, no se confunden los casos de las enfermedades Dermatitis seborreica y Pitiriasis rosada. En general, el procedimiento identifica correctamente los casos atípicos y típicos y además los clasifica correctamente.

Tabla 4.17: Estudio de la bondad del test INCA aplicado a los datos de diagnóstico dermatológico. Se han seleccionado al azar diferentes casos de los grupos previamente diagnosticados y se ha mirado si el test los identifica correctamente como típicos.

			Clasificación de los casos típicos					
Cluster	Casos testados	Casos detectados como típicos	1	2	3	4	5	6
1- Psoriasis	33	28	28	0	0	0	0	0
2- Dermatitis seborreica	25	25	0	24	0	1	0	0
3- Liquen plano	25	23	0	0	23	0	0	0
4- Pitiriasis rosada	10	10	0	0	0	10	0	0
5- Dermatitis crónica	8	8	0	0	0	0	8	0
6- Pitiriasis rubra pilaris	5	4	0	0	0	0	0	4

4.5.2. Datos de cáncer linfático

El objetivo del siguiente ejemplo es demostrar que los métodos desarrollados en este capítulo de la memoria son útiles para identificar situaciones en las que los

datos no presentan una clara estructura de clusters. Además, este ejemplo es útil para describir un nuevo procedimiento de cluster basado en el índice INCA. Los datos que se utilizan en este ejemplo son datos públicos cedidos por M. Zwitter y M. Soklic. Son datos relativos a un estudio sobre cáncer linfático y han sido obtenidos del Centro Médico Universitario, Instituto de Oncología, Ljubljana y que se pueden encontrar en la base de datos de UCI Knowledge Discovery (Hettich and Bay, 1999). Los datos consisten en 148 casos relacionados con el diagnóstico de cáncer linfático. Se incluyen cuatro tipos diferentes de diagnóstico: normal, metastasis, linfoma maligno y fibrosis con 2, 81, 61 y 4 casos, respectivamente. Obsérvese cómo para cualquier método será muy difícil determinar la existencia de 4 clusters debido al pequeño tamaño muestral de dos de los grupos. Sobre cada caso se han medido 18 variables mixtas: 1 cuantitativa (número de ganglios en la biopsia); 9 binarias (bloqueo de la aféresis, bloqueo de los cuerpos linfáticos, bloqueo de los senos linfoides, circulación aberrante, extravasación de linfas, regeneración, captación precoz, luxación y exclusión de ganglios); 8 categóricas (linfáticos, reducción del tamanño de los ganglios, agrandamiento del tamaño de los ganglios, cambios en la forma de los ganglios, defectos en el ganglio, cambios en el ganglio, cambios en la estructura del ganglio y formas especiales con 4, 4, 4, 4, 4, 4, 8 y 3 categorías, respectivamente). Dado que los datos son mixtos hemos utilizado la distancia de Gower, que ya fue definida en el capítulo 2 de la memoria.

Como en el ejemplo precedente, se ha utilizado el procedimiento PAM para obtener particiones de los 148 casos en 2, ..., 9 y 10 clusters. De todas maneras

se debe tener en cuenta que algún otro procediemiento de clasificación generaría particiones diferentes de estos mismos datos lo que podría llevar a diferentes estimaciones del número de clusters. En la Tabla 4.18 se muestran las particiones que ofrece PAM en 2, 3, 4 y 5 clusters. Cabe destacar que no se obtiene una clasificación clara para $k = 4$ (precisión del clustering 42.6 %). Probablemente se debería utilizar algún otro procedimiento de clustering o bien las variables consideradas en el estudio no discriminan adecuadamente los tipos de diagnósticos. En base a esta partición uno no esperaría encontrar ninguna estructura de clusters. La Figura 4.4 muestra los valores del índice INCA frente a k, para $k = 2, \ldots, 10$. Los bajos valores del índice señalan la falta de estructura de clusters de los datos, tal como se esperaba por las razones que acabamos de mencionar.

A continuación y en base a este ejemplo, se comenta cómo el test INCA puede ser útil en el desarrollo de un nuevo procedimiento de cluster basado en estrategias de aprendizaje. En primer lugar, puesto que $INCA_k$ ofrece una medida de la bondad de la partición considerada, se pueden reasignar los individuos hasta que se obtenga una mejor partición, es decir, hasta que $INCA_k$ alcance el valor 1 o bien hasta que este valor no se pueda mejorar. Así para $k = 2$ y realizando de esta forma reasignaciones de los casos se consigue que los casos de los grupos normal y fibrosis queden bien separados (ver Tabla 4.19). Estas consideraciones dan pie a considerar el test INCA como herramienta para introducir un nuevo procedimiento de clusters basado en estrategias de aprendizaje. Es decir, se considera una partición en k clusters obtenida utilizando un método cualquiera. Como primer paso se extraen algunos objetos de estos clusters formando así una

Tabla 4.18: Partición obtenida sobre los datos de cáncer linfático para diferentes valores de k utilizando el método PAM. Las particiones obtenidas se comparan con la clasificación real en 4 grupos de los datos.

	Clusters reales			
Clusters PAM	Normal	Metástasis	Linf. maligno	Fibrosis
C_1	2	26	29	1
C_2	0	55	32	3
C_1	2	23	26	0
C_2	0	24	23	4
C_3	0	34	12	0
C_1	2	21	12	0
C_2	0	24	16	4
C_3	0	33	9	0
C_4	0	3	24	0
C_1	2	11	4	1
C_2	0	23	16	3
C_3	0	28	8	0
C_4	0	16	9	0
C_5	0	3	24	0

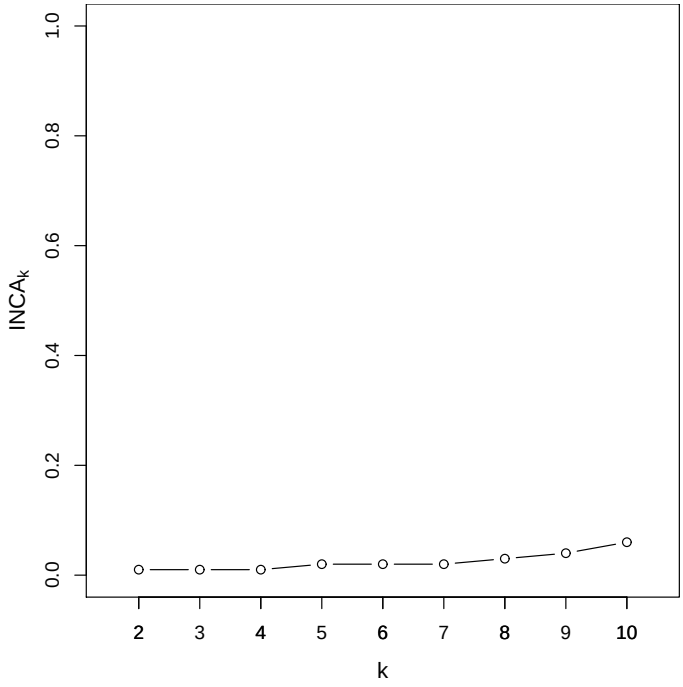


Figura 4.4: Estimación del número de clusters utilizando el índice INCA. Valores de $INCA_k$ frente a k aplicado a los datos del cáncer linfático. Obsérvese que no se detecta estructura de clusters.

partición en k clusters más pequeños. Esta extracción se realiza exigiendo que los nuevos clusters tengan una variabilidad geométrica pequeña y que la distancia entre ellos sea lo más grande posible. El segundo paso del método consiste en aplicar el test INCA a cada uno de los objetos no seleccionados en el primer paso. Aplicando este procedimiento a los datos de cáncer y $k = 2$, la clasificación obtenida después de unas pocas sesiones se muestran en la Tabla 4.20. Los subclusters extraídos en el primer paso no contenían casos de los grupos normal y fibrosis. Obsérvese que se han caracterizado como atípicos los dos casos del grupo

normal, los cuatro de fibrosis, cuatro de metástasis y dos de linfoma maligno. Estos primeros resultados parecen indicar la viabilidad y buenos resultados en la aplicación de esta metodología. Es evidente que es necesario un mayor trabajo al respecto sobre todo para mejorar las reglas de extracción de los sub-clusters iniciales.

Tabla 4.19: Grupos reales y clusters obtenidos con el método de reasignación aplicado a los datos de cáncer linfático y tomando $k = 2$.

	Clusters reales			
Clusters PAM	Normal	Metástasis	Linf. maligno	Fibrosis
C_1	2	24	31	0
C_2	0	57	30	4

Tabla 4.20: Grupos reales y clusters obtenidos con el método basado en técnicas de aprendizaje aplicado a los datos de cáncer linfático y tomando $k = 2$.

	Clusters reales			
Clusters PAM	Normal	Metástasis	Linf. maligno	Fibrosis
C_1		0	40	
C_2		77	19	

4.6. Conclusiones

En este capítulo hemos propuesto una metodología para estimar el número de clusters y para decidir si un nuevo objeto pertenece o no a un nuevo y desconocido cluster.

La metodología INCA presenta la ventaja de que puede aplicarse sobre cualquier tipo de datos y que puede ser aplicada sobre datos continuos sin necesidad de suponer ningún tipo de distribución sobre los mismos. El test INCA es una extensión de algunos métodos previamente desarrollados, presentando la novedad de que puede ser aplicado al caso de más de 2 clusters o incluso para cualquier tipo de datos. Se ha validado la metodología INCA utilizando diferentes y variados conjuntos simulados de objetos. De acuerdo con los resultados obtenidos, el índice INCA ofrece una herramienta útil en la estimación del número de clusters, siendo además robusto frente a situaciones de ruido. Las simulaciones también ofrecen buenos resultados del test INCA. Al aplicar esta metodología a datos reales, el índice INCA ha sido capaz de estimar correctamente el número de clusters en el caso de los diagnósticos de enfermedades eritemaescamosas y también ha sido capaz de detectar la ausencia de estructura de clusters en el caso de los datos de cáncer linfático. La eficacia del test INCA como herramienta para detectar objetos atípicos, también se ha demostrado con los datos de las patologías dermatológicas. Todos estos resultados parecen confirmar la utilidad de los procedimientos presentados en este capítulo, los cuales proporcionan una alternativa a otros métodos que imponen restricciones sobre el tipo de datos o el número de

grupos.

4.7. Difusión del método

Una primera aproximación del procedimiento para la detección de objetos atípicos se presentó en el *XXVIII Congreso Nacional de Estadística e Investigación Operativa*, 2004, celebrado en Cádiz. Así mismo, los primeros resultados sobre la estimación del número de clusters se presentó a la comunidad científica en el *25th European meeting of Statisticians*, 2005, que tuvo lugar en Oslo. Todos los resultados presentados en este capítulo de la memoria forman parte de un artículo, «INCA: New statistic for estimating the number of clusters and identifying atypical units», publicado en *Statistics in Medicine*, 27(15), 2948-2973, (2008).

4.8. Software desarrollado

En el paquete `DBmethods` se han incluido las siguientes funciones para poder ejecutar el método desarrollado:

- `dgower(x, type=list())`: dada una matrix de datos `x` e indicando las posiciones de columna de las variables cuantitativas, binarias y cualitativas en el parámetro `type`, calcula la matriz de distancias de Gower tal y como se describe en el capítulo 2. Cuando la matriz de datos `x` contiene algún valor

ausente para alguna observación, automáticamente se calcula la distancia de Gower para valores ausentes (Anexo A, pág. 233; Anexo B, pág. 276).

Esta función necesita las funciones auxiliares `gowerNOmissing` (Anexo B, pág. 290) y `gowerWITHmissing` (Anexo B, pág. 292).

- `estW(d, dx0, pert= "onegroup")`: dadas una matriz de distancias `d` correspondiente a los objetos de una partición de un conjunto C, la composición de dicha partición, `pert` y las distancias de un nuevo objeto $\mathbf{z}_0$ a los objetos de C, `dx0`, calcula la estimación del estadístico $W(\mathbf{z}_0)$ (Anexo A, pág. 237; Anexo B, pág. 280).

  Esta función necesita las funciones auxiliares `deltas_simple` (Anexo B, pág. 274) y `proxi_simple` (Anexo B, pág. 310).

- `INCAnumclu(d, pert_clus)`: dadas una matriz de distancias `d` correspondiente a los objetos de una partición en k de un conjunto C y la composición de dicha partición, `pert_clus`, calcula el índice $INCA_k$ (Anexo A, pág. 225; Anexo B, pág. 295).

  Esta función necesita las funciones auxiliares `deltas_simple`, `proxi_simple`, `estW_simple` (Anexo B, pág. 282), `maxW_k1` (Anexo B, pág. 302) y `maxW_k` (Anexo B, pág. 300).

- `INCAtest(d, pert, d_test, np=1000, alpha=0.05, P=1)`: dadas una matriz de distancias `d` correspondiente a los objetos de una partición en k de un conjunto C, la composición de dicha partición, `pert`, y las distancias `d_test` de un nuevo objeto $\mathbf{z}_0$ a los objetos de C, calcula el test de tipicalidad para $\mathbf{z}_0$. Por defecto calcula la distribución nula de $W(\mathbf{z}_0)$

sobre una muestra bootstrap de tamaño `np=1000`, establece el nivel de significación `alpha=0.05`, repetiendose el proceso $P = 1$ veces. Hay que hacer notar que esta función puede llevar bastante tiempo de ejecución ya que tiene que calcular la distribución nula del estadístico (Anexo A, pág. 227; Anexo B, pág. 298).

Esta función necesita las funciones auxiliares `deltas_simple`, `proxi_simple`, `estW_simple` y `distrW` (Anexo B, pág. 277).

Capítulo 5

Métodos de cluster basados en la denominada *path distance*

5.1. Introducción

Como se ha comentado en el capítulo 3 los métodos cluster divisivos, incluso cuando el número de objetos a clasificar es reducido, requieren de programas informáticos que consumen tiempos de ejecución extremadamente largos. Además es conocido que el uso de réplicas en la experimentación está cada vez más extendido. Hay que tener en cuenta que replicar no es sinónimo de encontrar copias exactas de las medidas realizadas, sea porque se haya producido algún tipo de error, sea porque los objetos en estudio sean muy sensibles a pequeños y a veces incontrolables cambios en el proceso experimental. Por ello y tal como se

ha comentado en la introducción de la memoria, pareció interesante plantear la posibilidad de desarrollar un método cluster divisivo que no tuviera ningún tipo de limitación en cuanto al número de objetos a clasificar y que además fuera aplicable a datos obtenidos en experimentos replicados. Para abordar pues este tipo de situaciones, en este capítulo se define una nueva distancia denominada *path distance*. Esta nueva distancia surgió del estudio de una situación hipotética como la que se muestra en la Figura 5.1. En esta figura se representan 7 objetos de forma que nadie dudaría en afirmar que los objetos forman dos clusters quedando agrupados los objetos 6 y 7 en un cluster y el resto en otro cluster, de manera que $C_1 = \{1, 2, 3, 4, 5\}$ y $C_2 = \{6, 7\}$. Sin embargo, en base a la distancia euclídea, el objeto 5 está tan cerca del objeto 1 como del objeto 7. De aquí se concluye que agrupamos de una manera natural los objetos 1 y 5 en el mismo cluster porque ambos tienen cerca objetos comunes. En base a esta idea intuitiva de agrupar objetos en un mismo cluster si existen otros objetos que de alguna forma los relacionan, se definió la que denominamos *path distance* y se desarrolló un nuevo algoritmo de cluster, tal como se describe a continuación.

Figura 5.1: Situación hipotética que motivó la definición de *path distance*.

5.2. Definiciones

Sea C un conjunto de n objetos con valores en un espacio métrico $\mathcal{R} \subset \mathbf{R}^p$ y función de densidad f respecto una medida adecuada. Supongamos que δ es una distancia definida sobre los objetos, cuyos valores están recogidos en una matriz D.

Denotaremos como $\{C_l : l = 1, 2\}$ una partición de C, donde cada cluster C_l tiene tamaño n_l $(l = 1, 2)$, respectivamente.

Definición 5.1.

Dados dos objetos i, j del cluster C_l, llamaremos path *entre i y j respecto de C_l, y lo denotaremos por $P_{ij}^{C_l}$ a una de las posibles formas de conectar i y j sin repetir ningún objeto.*

Ejemplo:

Sea $C = \{1, 2, 3, 4, 5\}$ un cluster de 5 objetos donde las distancias entre los 5 objetos están recogidas en la matriz D,

$$D = \begin{pmatrix} 0 & & & & \\ 0.3142 & 0 & & & \\ 0.3778 & 0.3633 & 0 & & \\ 0.3963 & 0.3449 & 0.4246 & 0 & \\ 0.4531 & 0.4095 & 0.5438 & 0.2875 & 0 \end{pmatrix}. \tag{5.1}$$

Fijemos, por ejemplo, los objetos 1 y 4. Existen 16 diferentes posibilidades de conectar 1 con 4 sin repetir ningún objeto, y por tanto diferentes *paths* entre 1 y 4:

$$\begin{array}{l} 1 \longrightarrow 4 \\ 1 \longrightarrow 2 \longrightarrow 4 \\ 1 \longrightarrow 3 \longrightarrow 5 \longrightarrow 4 \\ \vdots \end{array}$$

Definición 5.2.

Dados dos objetos i, j del cluster C_l y fijado un path $P_{ij}^{C_l}$ *entre i y j respecto de C_l, el* mayor paso *del* path *es el mayor valor de las distancias entre dos objetos del* path*,*

$$max\{\delta_{rs} \, : \, (r,s) \in P_{ij}^{C_l}\}.$$

Ejemplo (cont.):

Para los *paths* anteriormente descritos, se muestran a continuación las distancias δ_{ij} entre dos objetos consecutivos i y j, indicando en **negrita** el mayor paso de cada *path*.

$$\begin{array}{l} 1 \overset{\mathbf{0.3963}}{\longrightarrow} 4 \\ 1 \overset{0.3142}{\longrightarrow} 2 \overset{\mathbf{0.3449}}{\longrightarrow} 4 \\ 1 \overset{0.3778}{\longrightarrow} 3 \overset{\mathbf{0.5438}}{\longrightarrow} 5 \overset{0.2875}{\longrightarrow} 4 \\ \vdots \end{array}$$

Definición 5.3.

Dados dos objetos i, j del cluster C_l consideremos todos los posibles paths *que conectan i y j. La* path distance *entre i y j respecto de C_l, $dp_{C_l}(i,j)$, es el mínimo de todos los mayores pasos encontrados para todos los posibles* paths.

$$dp_{C_l}(i,j) = min_{P_{ij}^{C_l}}\{max(\delta_{rs} \,:\, (r,s) \in P_{ij}^{C_l})\}.$$

Luego, dentro del cluster C_l, los objetos i y j están conectados mediante pasos de distancias menores o iguales que $dp_{C_l}(i,j)$.

Ejemplo (cont.):

Teniendo en cuenta los 16 *paths* que hay en C que conectan 1 y 4, el menor de los mayores pasos es 0.3449, justamente el dado por $1 \overset{0.3142}{\longrightarrow} 2 \overset{\mathbf{0.3449}}{\longrightarrow} 4$, y por tanto, $dp_C(1,4) = 0.3449$.

En la Figura 5.2 se muestran los valores de todos los conceptos introducidos en las definiciones anteriores para los datos del ejemplo.

Definición 5.4.

Dado un cluster C_l, la path distance *de C_l es el máximo de las* path distances *entre todos los pares de objetos i, j de C_l*

$$dp(C_l) = max_{i,j \in C_l}\{dp_{C_l}(i,j)\}.$$

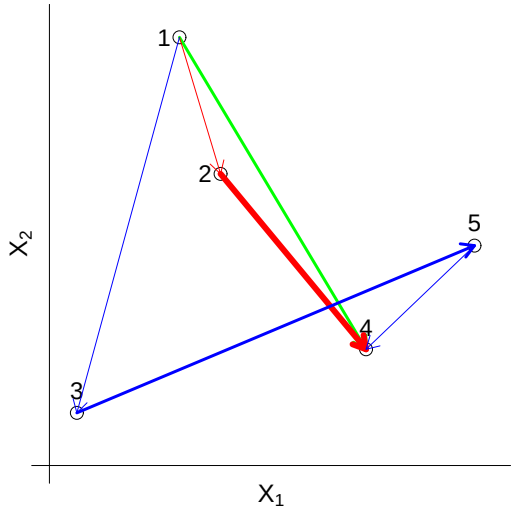


Figura 5.2: Representación de los objetos del cluster del ejemplo $C = \{1, 2, 3, 4, 5\}$ mediante las dos primeras coordenadas principales calculadas sobre la matriz de distancias (5.1). Las flechas indican los *paths* $1 \longrightarrow 4$, $1 \longrightarrow 2 \longrightarrow 4$ y $1 \longrightarrow 3 \longrightarrow 5 \longrightarrow 4$. En cada caso la línea más gruesa indica el mayor paso del *path*. La flecha roja y gruesa que une 2 con 4, indica el menor de los mayores pasos de todos los 16 *paths* que unen 1 y 4 y corresponde por tanto a la *path distance* $dp_C(1, 4)$.

Ejemplo (cont.):

Si se computan todos los posibles valores de este ejemplo, se obtiene que $dp(C) = 0.3633$, justamente la *path distance* entre 2 y 3. Dicho de otra manera, partiendo de cualquier objeto de C se puede llegar a cualquier otro en pasos menores o iguales a 0.3633.

Hay que hacer notar que como consecuencia de las definiciones anteriores la *path distance* de un cluster es también una medida de su variabilidad.

Una vez introducida esta nueva distancia, vamos a centrarnos en el caso de los experimentos replicados. Por lo tanto sea C un conjunto de n objetos representados por el p-vector aleatorio $\mathbf{Z}$ de densidad f respecto de una medida adecuada y soporte $\mathcal{R}$. Sea, como es habitual, δ una distancia entre los objetos $\mathbf{z}_i$, $i = 1, \ldots, n$. Supongamos que la medición del valor de las variables sobre estos objetos se ha replicado, es decir, hemos repetido R veces el experimento obteniendo para cada objeto y cada variable R vectores, $\mathbf{z}_{i,r}$, $i = 1, \ldots, n$, $r = 1, \ldots, R$. Por tanto, en esta situación se tienen R matrices $\mathbf{Z}_r$ $(n \times p)$, una por cada réplica, y cada una de ellas conteniendo las medidas de los n objetos para las p variables consideradas.

Al realizar diferentes medidas sobre el objeto (réplicas) es de esperar que se encontrará cierta variabilidad, por ello se considerará el siguiente concepto, que en el caso particular de trabajar con una única variable observable ($p = 1$) continua y utilizar la distancia euclídea coincidirá con el concepto de desviación típica entre las réplicas.

Definición 5.5.

Sea i un objeto y $\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,R}$ los R vectores que contienen los valores de las p variables observables sobre las R réplicas. Sea $\delta_{rs,i}$ la distancia entre las réplicas r y s del objeto i. Se define la variabilidad *DV_i del objeto i para las R réplicas como la raíz cuadrada de la variabilidad geométrica entre las R réplicas obtenidas sobre el objeto i,*

$$DV_i = V_i^{1/2} = \left[\frac{1}{2R^2} \sum_{r,s=1}^{k} \delta_{rs,i}^2 \right]^{1/2}.$$

Antes de pasar a desarrollar el nuevo algoritmo, es necesario introducir una nueva definición.

Definición 5.6.

Dado un conjunto C de objetos y un valor umbral α, dos clusters C_1 y C_2, $C_1 \cap C_2 = \emptyset$, $C_1 \cup C_2 = C$, definen una partición de C factible *con λ y α si se verifican las dos condiciones siguientes:*

1. $dp(C_1) \leq \lambda$ *y* $dp(C_2) \leq \lambda$.
2. $\max\{|DV_i - DV_j| : i, j \in C_s\} \leq \alpha$, $s = 1, 2$.

La condición 1 pretende encontrar clusters compactos en base a la *path distance* y la condición 2 pretende encontrar clusters homogéneos en términos de

variabilidad entre réplicas. Es decir, particiones que cumplan ambas condiciones serán particiones que agrupen objetos tanto en relación a su distancia como en cuanto a la variabilidad entre sus réplicas, buscando clusters que separen a los objetos con menor y mayor variabilidad entre réplicas respectivamente.

La condición 2 puede reformularse de la siguiente forma.
Sean $v_{min} = \text{mín}\{DV_i : i \in C\}$ y $v_{max} = \text{máx}\{DV_i : i \in C\}$ las variabilidades mínima y máxima de los objetos de C, respectivamente. Puesto que (C_1, C_2) define una partición de C, la condición 2 es equivalente a:

2'. $DV_i \in [v_{min}, v_{min} + \alpha]$, $\forall i \in C_1$ y $DV_i \in [v_{max} - \alpha, v_{max}]$, $\forall i \in C_2$.

En efecto, supongamos que $v_{min} = DV_i$ de manera que $i \in C_1$.

Entonces, $DV_j - v_{min} \leq \alpha$, $\forall j \in C_1$. Es decir,

$$DV_j \leq \alpha + v_{min}.$$

Evidentemente $v_{min} \leq DV_j$, por lo que se concluye que

$$v_{min} \leq DV_j \leq v_{min} + \alpha \quad \forall j \in C_1.$$

Sea $v_{max} = DV_{i'}$ tal que $i' \in C_1$. En este caso $v_{max} - v_{min} \leq \alpha$ por lo que en realidad se tiene que $|DV_j - DV_k| \leq \alpha$ para cualesquiera objetos j, k y la condición 2 no añade nada nuevo. Por tanto, podemos restringirnos al caso en que $v_{max} = DV_{i'}$ tal que $i' \in C_2$ ($C_1 \cap C_2 = \emptyset$). Luego $v_{max} - DV_j \leq \alpha$, $\forall j \in C_2$ y

$$DV_j \geq v_{max} - \alpha.$$

Puesto que $v_{max} \geq DV_j$, se concluye que

$$v_{max} - \alpha \leq DV_j \leq v_{max} \quad \forall j \in C_2.$$

5.3. Metodología

Dado un conjunto de objetos C y fijado un valor umbral α, el algoritmo que se propone a continuación tiene por objetivo encontrar el valor mínimo de λ para el que existe un partición de C de forma que sea una partición factible para λ y α. La obtención de una tal partición requiere de un proceso que podemos separar en dos etapas.

Etapa 1.

Una vez fijados los valores de α y λ, es fácil saber si existen dos clusters C_1 y C_2, $C_1 \cap C_2 = \emptyset$, $C_1 \cup C_2 = C$, que formen una solución factible para λ y α. En efecto, sea $i_0 \in C$ un objeto cualquiera de C seleccionado de forma arbitraria pero verificando que $DV_{i_0} \in [v_{min},\, v_{min} + \alpha]$. Se define un cluster A de forma que inicialmente sólo contiene este objeto i_0. A continuación se clasifican en A todos los objetos i de C cuya distancia a i_0 sea menor o igual que λ y que verifiquen que $DV_i \in [v_{min},\, v_{min} + \alpha]$. Este proceso se repite de manera que se van incluyendo en A todos los objetos de C con variabilidad en el intervalo $[v_{min},\, v_{min} + \alpha]$ y que están a una distancia menor o igual que λ de cualquier objeto previamente asignado en A. Cuando no hay más objetos en C que puedan ser incluidos en A termina este proceso. Este proceso nos garantiza que $dp(A) \leq \lambda$, ya que para

cualquier par de objetos en A existe un *path* cuyo paso mayor es menor o igual a λ.

Si $A = C$, supongamos que j designa el último objeto incluido en A. Entonces, los clusters $C_1 = A \setminus \{j\}$, $C_2 = \{j\}$ definen una partición de C factible para λ y α.

Si no fuera así, consideremos $C_1 = A$. Ahora se define un nuevo cluster B de manera similar al proceso anterior pero restringido a los objetos de $C \setminus C_1$ con variabilidad en $[v_{max} - \alpha, v_{max}]$. Es decir, se selecciona como primer objeto de B un objeto arbitrario de $C \setminus C_1$ con variabilidad en $[v_{max} - \alpha, v_{max}]$. A continuación, se incluyen en B todos los objetos de $C \setminus C_1$ con variabilidad en $[v_{max} - \alpha, v_{max}]$ y que se encuentren a una distancia menor o igual que λ de cualquier objeto previamente incluido en B. Se repite este proceso hasta que no se pueda incluir en B ningún objeto más.

Si $B = C \setminus C_1$, entonces, los clusters C_1 y $C_2 = C \setminus C_1$ definen una partición de C factible para λ y α. En otro caso no existe una partición de C factible con λ y α.

De este modo y para valores fijados de λ y α, diremos que la respuesta de este procedimiento es SÍ cuando existe una partición en C_1 y C_2 factible para λ y α. Diremos que la respuesta de este procedimiento es NO cuando no existe tal partición.

Etapa 2. Búsqueda binaria

Dado un conjunto C y fijado un valor umbral α, λ^* denota la solución óptima del problema de optimización consistente en encontrar el mínimo valor de λ para el que existe una partición en dos clusters C_1 y C_2 factible para λ y α. El algoritmo que se propone a continuación encuentra este valor óptimo aplicando lo que se conoce como algoritmo de *búsqueda binaria*. Éste es un algoritmo iterativo en el que al inicio de cada iteración se conoce un intervalo que contiene a λ^* y este intervalo es reducido a la mitad en cada iteración. Para dividir el intervalo el razonamiento que se sigue es el siguiente. Sea $[a_t, b_t]$ el intervalo que contiene el valor λ^* al inicio de la t-ésima iteración y consideramos el valor $\lambda_t = (a_t + b_t)/2$. Después de aplicar el procedimiento descrito en la etapa 1 para los valores λ_t y α, si la respuesta es SI, entonces a la partición óptima no le puede corresponder un valor superior a λ_t. Por tanto, se puede reducir el intervalo a $[a_t, \lambda_t]$. En caso contrario, si la respuesta es NO, sabemos que a la partición óptima le corresponde un valor superior a λ_t y como consecuencia podemos reducir el intervalo a $[\lambda_t, b_t]$. Además, puesto que el valor λ^* debe ser un valor de la matriz de distancias D, cuando la respuesta es SI, el extremo superior del intervalo $[a_t, \lambda_t]$ puede ser reemplazado por $\text{máx}\{\delta_{ij} \,:\, \delta_{ij} \leq \lambda_t \,,\, i,j \in C\}$. De manera similar, cuando la respuesta es NO, el extremo inferior del intervalo puede ser reemplazado por $\text{mín}\{\delta_{ij} \,:\, \delta_{ij} \geq \lambda_t \,,\, i,j \in C\}$. Repitiendo este procedimiento con el nuevo intervalo, el algoritmo finaliza cuando $a_t = b_t$, y la partición óptima es la correspondiente a la última vez en la que la respuesta fue SI. Para la primera iteración se considera el intervalo dado por $a_1 = \text{mín}_{i,j \in C}\, \delta_{ij}$ y $b_1 = \text{máx}_{i,j \in C}\, \delta_{ij}$.

Así pues, el algoritmo divisivo propuesto se puede formalizar de la forma siguiente:

Inicializar: $t = 1$, $a_1 = \text{mín}_{i,j \in C}\, \delta_{ij}$, $b_1 = \text{máx}_{i,j \in C}\, \delta_{ij}$

Mientras $a_t \neq b_t$ *hacer iteraciones*

En cada una de ellas:

$\lambda_t := (a_t + b_t)/2$

Aplicar procedimiento Etapa 1 con el valor λ_t

Si respuesta SI *entonces*

$a_{t+1} = a_t$

$b_{t+1} = \text{máx}\{\delta_{ij} \,:\, \delta_{ij} \leq \lambda_t, i, j \in C\}$

Anotar C_1 y C_2

$\lambda^* = \lambda_t$

en caso contrario

$a_{t+1} = \text{mín}\{\delta_{ij} \,:\, \delta_{ij} \geq \lambda_t, i, j \in C\}$

$b_{t+1} = b_t$

Terminar condicional

Pasar a la siguiente:

$t := t + 1$

Acabar cuando a_t=b_t

Observaciones:

Como se comenta a continuación, hay casos particulares en los que el problema de optimización no es factible o bien puede ser resuelto por otros medios.

- Cuando el valor de α es tal que existe $i \in C$ tal que $v_{min} + \alpha < DV_i <$

$v_{max} - \alpha$, no existe ningún valor λ para el que la partición sea factible para λ y α y por tanto el problema de optimización no es factible.

- Cuando el problema es factible y $v_{min} + \alpha \leq v_{max} - \alpha$, el problema de optimización es trivial ya que los dos intervalos $[v_{min}, v_{min} + \alpha]$ y $[v_{max} - \alpha, v_{max}]$ definen la única partición de C factible para λ y α.

- Cuando el valor de α es tal que $v_{min} + \alpha \geq v_{max}$, la condición 2 de la Definición 5.6, es redundante y la partición será factible o no únicamente en función de λ. En este caso particular, el algoritmo presentado utilizaría como única condición la dada por 1 en la Definición 5.6, y por tanto se basaría únicamente en minimizar las correspondientes *path distances*. En tal caso, el algoritmo presentado coincidiría con el método MST del minimum spanning tree (Xu *et al.*, 2002; Varma and Simon, 2004). En particular, el valor del mínimo de las *path distances* es el valor de la mayor arista de MST. Sin embargo, hay que observar que cuando la restricción 2 de la Definición 5.6 no es redundante y los intervalos $[v_{min}, v_{min} + \alpha]$ y $[v_{max} - \alpha, v_{max}]$ se solapan, el método MST no puede resolver el problema de optimización ya que éste no garantiza que la (raíz cuadrada de la) variabilidad geométrica de los objetos de un mismo subárbol estén dentro del mismo intervalo.

- El algoritmo encuentra el mínimo global para λ^* aunque no se puede asegurar que la partición encontrada sea única. En el caso en que $A = C$, la partición que se obtiene dependerá de los parámetros iniciales, ya que el objeto asignado a C_2 solamente depende del orden en que los objetos son añadidos a A. En el caso en que $v_{max} - \alpha \leq v_{min} + \alpha$, el algoritmo

devuelve el mayor cluster posible para C_1. No se presenta ningún tipo de discusión sobre cuándo una partición es la mejor, ya que si existe más de una posible solución sería necesario añadir alguna nueva condición a fin de poder seleccionar cuál se consideraría como la mejor de todas las particiones obtenidas.

Por todo lo comentado anteriormente, se recomienda pues fijar el valor de α como $\alpha = A\,(v_{max} - v_{min})$ de manera que $1/2 < A \leq 1$.

Finalmente, cabe comentar que a la hora de interpretar los resultados de este algoritmo es de mucha ayuda visualizarlos gráficamente. Cuando el algoritmo se aplica a datos relativos a experimentos sin réplicas o bien se selecciona $\alpha = v_{max} - v_{min}$, el algoritmo produce sucesivas divisiones donde cada división tiene asociado el valor λ de manera que $dp(C_l) \leq \lambda$ $(l = 1, 2)$. Por tanto, representaremos las sucesivas divisiones que ofrece el algoritmo en una estructura de árbol donde la altura de las aristas de una partición viene dada por el valor de *path distance* asociado a cada cluster de la división. Este procedimiento generará una ultramétrica ya que considerando una partición cualquiera C_1 y C_2 de las sucesivas divisiones que se dan en el desarrollo del algoritmo, ésta será factible para λ, con $dp(C_1) \leq \lambda$ y $dp(C_2) \leq \lambda$ y necesariamente $dp(C_1 \cup C_2) \geq \lambda$. Sin embargo, la jerarquía que se obtiene no será necesariamente indexada cuando el experimento tenga réplicas y $\alpha = A\,(v_{max} - v_{min})$ con $1/2 < A \leq 1$.

5.4. Aplicación a datos simulados

A continuación se presentan varios ejemplos de la utilización del procedimiento sobre datos simulados. El objetivo de estos ejemplos es estudiar el comportamiento del algoritmo en caso concretos. Por ejemplo, cuál es su comportamiento ante la presencia de clusters elípticos, si presenta o no el efecto encadenamiento y si es útil ante datos dispersos. Finalmente se presentan dos ejemplos ficticios con réplicas a fin de clarificar el funcionamiento y utilidad del método en el caso replicado.

1. *Clusters elípticos.*

   Se ha partido de una situación análoga a la descrita en la sección 3.2.3 del capítulo 3 y se han simulado 20 conjuntos de objetos agrupados en 2 clusters elípticos bien diferenciados como el representado en la Figura 5.3 (Izquierda). Tras aplicar el algoritmo se obtuvo que para 4 de los 20 conjuntos de objetos, el algoritmo identificó correctamente los dos clusters. En 8 de los 20 conjuntos de objetos, un único elemento quedó mal identificado. En concreto, si C representa el conjunto inicial de objetos a clasificar y C_1 y C_2 ($C = C_1 \bigcup C_2$) los dos grupos generados, en el primer paso del algoritmo se obtuvo un cluster con un único objeto $\{i\}$, y un cluster con el resto $C \setminus \{i\}$. En la siguiente partición, este segundo cluster da lugar a dos nuevos clusters quedando los objetos bien separados en relación a sus grupos iniciales, tal como se representa en la Figura 5.3 (Derecha). Finalmente, en otros 5 conjuntos de objetos se encontraron entre 3 y 6 individuos

mal identificados y para los 4 conjuntos de objetos restantes no fue posible encontrar de ninguna forma los dos grupos iniciales de objetos.

2. *El efecto encadenamiento.*

   Se estudió si el algoritmo presenta el denominado efecto encadenamiento y para ello se consideró la misma simulación que la descrita en la sección 3.2.3 del capítulo 3. Se generaron 20 objetos de cada cluster y se observó que en todas los conjuntos de objetos creados, el algoritmo no identificaba los clusters de forma correcta. Este resultado era de prever ya que la distancia en la que se basa el método, *path distance*, justamente considera cercanos dos objetos siempre que haya otros objetos intermedios y cercanos a ambos.

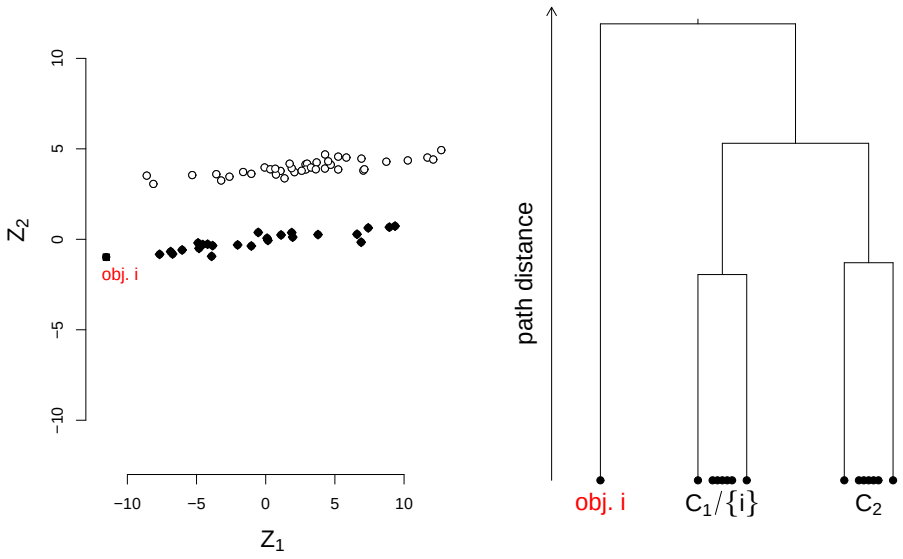


Figura 5.3: (Izquierda) Representación gráfica de uno de los 20 conjuntos de objetos simulados, que pertenecen a dos grupos C_1 (puntos en negrita) y C_2 (puntos blancos) bien diferenciados. (Derecha) Árbol obtenido al aplicar el algoritmo de cluster. Obsérvese que el objeto identificado por i del grupo C_1 (puntos en negrita) queda separado en el primer paso de la partición. En el segundo paso quedan bien separados el resto de objetos de C_1 de los de C_2.

3. *Datos* dispersos*(scattered points).*

   Como es conocido, se entiende por objetos dispersos aquellos que se encuentran dispersos entre clusters bien definidos. Para medir las posibilidades de nuestro método bajo esta situación se procedió como en Tseng (2005) simulando (ver Figura 5.4) 10 clusters bi-dimensionales, bajo la distribución normal, con matriz de varianzas-covarianzas 0.1^2I, 0.2^2I,...,1^2I, respectivamente, donde I designa la matriz identidad. El centro de cada cluster se eligió de forma aleatoria dentro del cuadrado $(0, 50) \times (0, 50)$. Todos los clusters estaban formados por 25 objetos de manera que cada objeto estuviera a una distancia respecto del centro del cluster al que pertenece menor a dos veces la desviación estándar. Es decir, si un objeto se encontraba respecto del centro del cluster al que pertenecía, a una distancia superior a dos veces la desviación estándar, se eliminaba del conjunto de datos. Además, se añadieron 30 objetos dispersos que fueron obtenidos a partir de una distribución uniforme en el cuadrado $(0, 50) \times (0, 50)$. Estos nuevos objetos se tomaron de forma que todos se encontraban con respecto a los centros de cualquiera de los 10 clusters, a una distancia superior a tres veces la correspondiente desviación estándar. De esta forma se generaron 20 conjuntos de objetos y se tomó como distancia inicial la distancia euclídea, calculando a partir de ella las correspondientes *path distances*. En este caso, los resultados fueron altamente satisfactorios ya que en los 20 conjuntos de datos generados se identificaron de forma correcta los 10 clusters y todos los objetos *dispersos*.

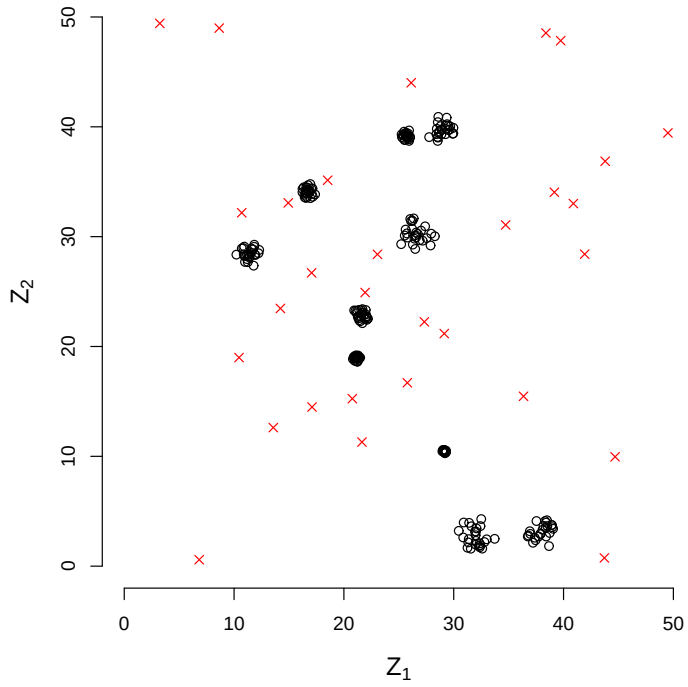


Figura 5.4: Posición de 10 clusters bien separados con 30 objetos *dispersos* (indicados por $\times$ en color rojo).

4. *Experimento replicado: datos continuos.*

En primer lugar se simuló un experimento con tres réplicas para $n = 11$ objetos. Una situación de este tipo podría encontrarse, por ejemplo, en un experimento en el que se midiera el nivel de expresión de 11 genes en una placa de microarrays. Las réplicas podrían representar mediciones independientes sobre la misma placa (en este caso las variaciones de los niveles se deberían a diferencias en los niveles de expresión del gen), o bien, mediciones de los mismos genes sobre diferentes placas (por tanto las diferencias en el nivel de expresión se deberían a errores experimentales). En este contexto, se supuso como es habitual que el nivel de expresión de un gen i para una réplica r $(i = 1, \ldots, 11;\ r = 1, \ldots, 3)$ sigue una distribución $N(\mu_i, \sigma_i^2)$. Los valores de μ_i se obtuvieron aleatóriamente a partir de una distribución uniforme $U(0, 110)$. A fin de generar genes con diferente variabilidad entre réplicas, los valores de σ_i se obtuvieron aleatóriamente utilizando las distribuciones $U(0, 0.0001)$ y $U(5, 10)$ para 5 y 6 genes respectivamente. En la Tabla 5.1 se detallan los valores generados.

Se consideró la distancia euclídea para medir la distancia entre pares de objetos. La partición que se obtuvo aplicando el algoritmo con $\alpha = 2/3(v_{max} - v_{min})$ se muestra en la Figura 5.5. Para simplificar, en la Figura no se muestran las particiones cuando un cluster tiene únicamente dos objetos. Los clusters obtenidos son:

$C_1 = \{1, 2, 3, 7, 8, 9, 10, 11\}$, $C_2 = \{4, 5, 6\}$, $C_3 = \{1, 3, 8, 9, 11\}$,

$C_4 = \{2, 7, 10\}$, $C_5 = \{4, 5\}$, $C_6 = \{6\}$, $C_7 = \{1, 3, 9\}$, $C_8 = \{8, 11\}$, $C_9 = \{2\}$, $C_{10} = \{7, 10\}$, $C_{11} = \{9\}$, $C_{12} = \{1, 3\}$.

Tabla 5.1: Datos continuos simulando un experimento con 3 réplicas.

Gen	Réplica 1	Réplica 2	Réplica 3	Media	Desv. estándar
1	3.88	3.78	5.53	4.40	0.98
2	3.01	7.34	1.40	3.92	3.07
3	17.36	16.64	18.86	17.62	1.13
4	14.10	20.28	10.31	14.90	5.03
5	8.32	10.35	1.56	6.74	4.60
6	33.65	37.11	44.80	38.52	5.71
7	37.66	42.06	43.47	41.06	3.03
8	44.98	44.83	45.64	45.15	0.43
9	44.97	42.13	42.86	43.32	1.47
10	45.56	45.45	50.31	47.11	2.78
11	60.72	60.25	59.93	60.30	0.40

Hay que destacar, tal y como se ha explicado en el algoritmo, que se tiene en cuenta no sólo el valor medio de los niveles de expresión para un cluster sino también la variabilidad entre sus réplicas. Por ejemplo, la primera división separa por una parte los genes con mayor variabilidad (cluster $C_2 = \{4, 5, 6\}$) del resto. En este paso, en cuanto a los niveles de expresión, todavía los clusters obtenidos son bastantes heterogéneos tal y como se refleja en que la *path distance* $dp(C_2) = 43.07$ y $dp(C_1) = 40.79$. Por otra parte, el algoritmo también tiene en cuenta los niveles medios de expresión. Por ejemplo, al considerar el cluster $C_4 = \{2, 7, 10\}$ y su división resultante en los clusters $C_9 = \{2\}$ y $C_{10} = \{7, 10\}$, se observa que aunque los genes 2

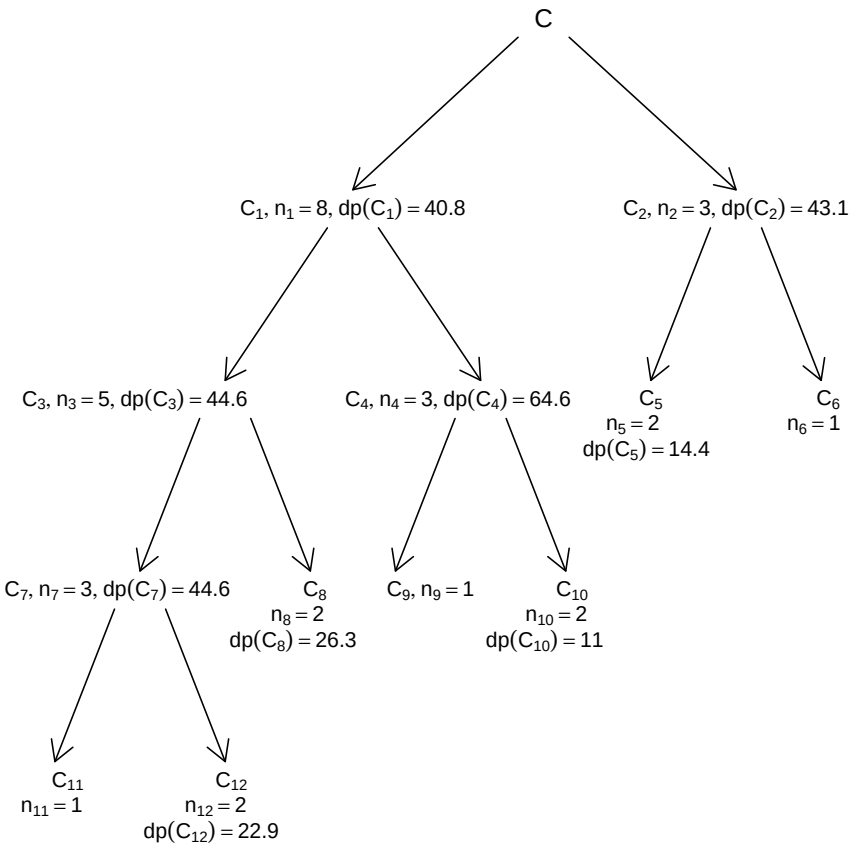


Figura 5.5: Representación gráfica de los clusters obtenidos sobre un conjunto de objetos simulados con 3 réplicas, suponiendo datos continuos. En cada paso se muestra el número de objetos de cada cluster y la correspondiente *path distance*.

y 7 tienen una variabilidad entre réplicas muy similar, estos dos genes son separados porque sus respectivos niveles de expresión son muy diferentes. Hacer la clasificación teniendo en cuenta estos dos aspectos puede ser muy

útil ya que permite determinar qué genes presentan una variabilidad entre réplicas similar y también a qué nivel de expresión medio son funcionales.

5. *Experimento replicado: caso general.*

   En este caso se simuló una situación en la que se midieran variables no necesariamente continuas y con réplicas. En particular, se construyeron dos clusters bien separados, pero de manera que en uno de ellos hubiera una mayor variabilidad entre las réplicas. Cada cluster se generó de tamaño 20 y como frecuencias obtenidas de una distribución multinomial de la siguiente manera: Cluster $C_1 \sim M(100, \pi_1 = 0.8, \pi_2 = 0.1, \pi_3 = 0.1)$ (objetos $1 - 20$) y $C_2 \sim M(10, \pi_1 = 0.1, \pi_2 = 0.5, \pi_3 = 0.4)$ (objetos $21 - 40$). Se generaron $R = 3$ réplicas de cada objeto. Nótese que al considerar la distribución multinomial del cluster C_2 sobre 10 «tiradas», la variabilidad entre las réplicas de ésta será mayor. Se ha elegido la distancia de Bhattacharyya como distancia entre pares de objetos y también entre pares de réplicas correspondientes a un mismo objeto.

   En este momento queremos destacar el papel que puede jugar el parámetro α. En la Figura 5.6 se muestran las soluciones relativas a los valores $\alpha = 4/5(v_{max} - v_{min})$ y $\alpha = 2/3(v_{max} - v_{min})$. Para $\alpha = 4/5(v_{max} - v_{min})$, la partición correcta se alcanza en la primera división. Para $\alpha = 2/3(v_{max} - v_{min})$, la primera partición ($\widetilde{C_1} = \{1, \ldots, 20, 21, 22, 23, 24, 25, 27, 28, 30, 31, 37, 38, 39\}$ y $\widetilde{C_2} = \{26, 29, 32, 33, 34, 35, 36, 40\}$) separa un cluster ($\widetilde{C_2}$) con los objetos de mayor variabilidad entre las réplicas, y el resto de los objetos que provienen del

segundo cluster son arrastrados a $\widetilde{C_1}$. Esto se debe a que al haber objetos i del segundo cluster de manera que $DV_i \in [v_{min},\, v_{min}+\alpha]$, estos objetos son forzosamente asignados al cluster $\widetilde{C_1}$ por la condición 2 de la Definición 5.6 y actúan como objetos intermedios a la hora de asignar el resto de objetos provenientes del segundo cluster al cluster $\widetilde{C_1}$.

El valor de α lo decidirá en cada caso el investigador de manera que cuando predomine la necesidad de identificar genes con gran variabilidad entre réplicas fijará un valor pequeño para α y en caso contrario, un valor grande.

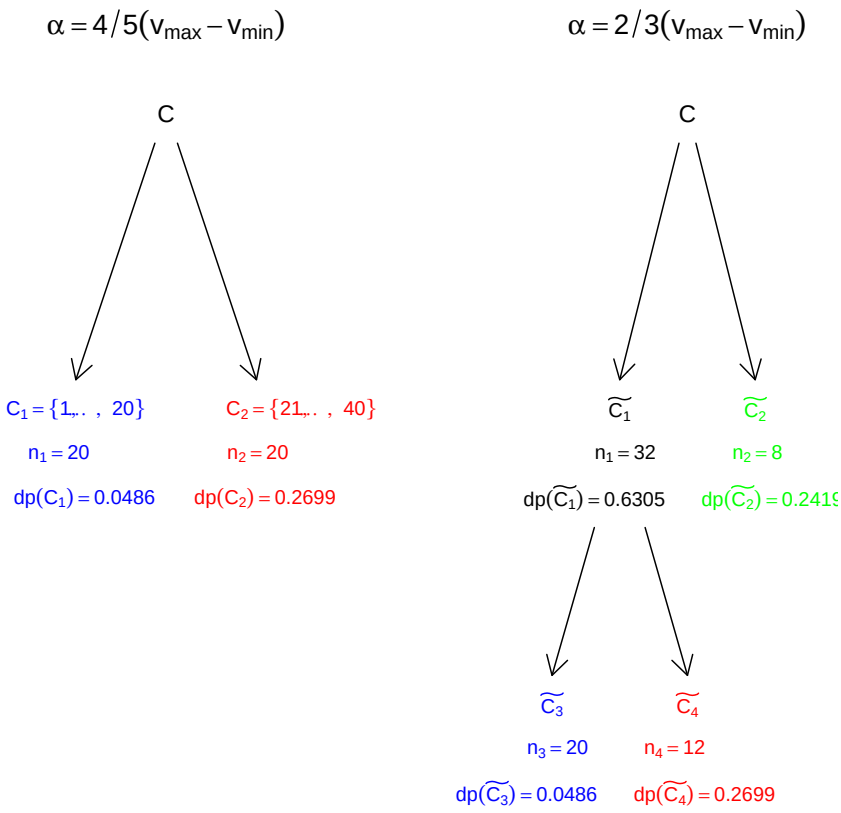


Figura 5.6: (Izquierda) Representación gráfica de la partición obtenida para el valor $\alpha = 4/5(v_{max} - v_{min})$ sobre un conjunto de objetos simulados con 3 réplicas y datos no continuos. (Derecha) Representación gráfica de las particiones obtenidas para el valor $\alpha = 2/3(v_{max} - v_{min})$ sobre un conjunto de objetos simulados con 3 réplicas y datos no continuos. En ambos casos se muestra el número de objetos de cada cluster y la correspondiente *path distance*.

5.5. Aplicación a datos reales

En esta sección se presenta la aplicación del método a conjuntos de datos reales relativos a la clasificación molecular de la leucemia (Golub *et al.*, 1999) y al estudio de los mecanismos funcionales del colesterol HDL (Callow *et al.*, 2000).

1. *Clasificación molecular de la leucemia.*

   La leucemia es uno de los tipos de cáncer más común. Mediante diferentes procedimientos de diagnóstico las leucemias agudas se pueden clasificar en dos subgrupos: las leucemias agudas linfoblásticas (ALL) y las leucemias agudas mieloides (AML). Aunque esta distinción ha sido bien establecida, aún no existe un único test sencillo que permita establecer el diagnóstico. Actualmente el diagnóstico lo establece un hematopatólogo experimentado que ha de ser capaz de interpretar la morfología tumoral (Frei *et al.*, 1961), la histoquímica (Bennett and Dutcher, 1969), el fenotipo inmunológico (Schlossman *et al.*, 1976) y el análisis citogenético (Golub *et al.*, 1995). Normalmente todas estas pruebas se efectúan en laboratorios especializados diferentes. Distinguir entre ALL y AML es fundamental para aplicar un tratamiento adecuado de quimioterapia (Pui and Evans, 1998). Aunque normalmente el diagnóstico del tipo de leucemia acostumbra a ser correcto, su clasificación continúa siendo imperfecta e incluso se producen errores. En el presente ejemplo, se han utilizado los datos de Golub *et al.* (1999) que consistían en 38 muestras de médula ósea procedentes de pacientes con leucemia aguda. Por los procedimientos de diagnóstico previamente comen-

tados se sabía que 27 eran del grupo ALL y 11 del AML. El RNA obtenido de las células de la médula ósea se hibridó con microarrays preparados por Affymetrix que contenían sondas de genes humanos. En el presente estudio las 27 ALL corresponden a los objetos 1 a 27 y las 11 muestras AML a los objetos 28 a 38.

Los datos se resumen en una matriz 38×3051 donde la entrada de la fila i y la columna j representa el nivel de expresión para el gen j en el mRNA de la muestra i. No hay valores faltantes o missings y los datos han sido estandarizados como se explica y justifica en el trabajo de Yang *et al.* (2002). Se ha utilizado la distancia correlación (ver (2.2) en el capítulo 2) para medir la distancia entre pares de muestras. Los resultados muestran cierto efecto de encadenamiento sobre 5 muestras de tipo ALL y otras 5 muestras de tipo AML. Sin embargo el algoritmo desarrollado resulta efectivo ya que detecta el cluster determinado por el resto de las muestras de tipo AML.

2. *Estudio de los mecanismos funcionales del colesterol HDL.*

Durante más de cincuenta años se ha reconocido la fuerte relación inversa entre el colesterol HDL (High Density Lipoprotein) y la susceptibilidad a la arterioesclerosis. Sin embargo los mecanismos subyacentes a su función todavía no están bien establecidos. Recientemente se han producido algunos avances gracias al uso de ratones transgénicos y *knockout*. Por ejemplo, los ratones *knockout* para la apolipoproteína AI (ApoAI) presentan niveles extraordinariamente bajos de colesterol HDL (Williamson *et al.*, 1992; Plump *et al.*, 1996). En el presente ejemplo se han utilizado los datos de Callow

et al. (2000), quienes analizaron mediante microarrays la expresión génica tanto en el hígado de ratones *knockout* para el gen que controla la ApoAI como en el de ratones control. Con ello querían obtener nueva información acerca de los efectos de la ApoAI sobre la expresión de otros genes a nivel del hígado.

A fin de ilustrar el método propuesto hemos considerado un subconjunto de los datos de Callow *et al.* (2000). Para ello se tomaron los primeros 50 genes, con tres réplicas cada uno, que se expresan diferencialmente de acuerdo con el estadístico t presentado en Callow *et al.* (2000). Uno de los genes, en concreto el gen 2, presentaba un valor de desviación estándar respecto la media de las réplicas, muy elevado (0.95) con respecto al resto (máximo valor = 0.58) por lo que fue eliminado del estudio. El algoritmo (con $\alpha = 2/3(v_{max} - v_{min})$) da lugar en un primer paso a dos clusters (ver Figura 5.7), uno incluyendo los genes con gran varianza entre las réplicas (cluster C_2) incluyendo 7 genes y otro grupo (cluster C_1) con los 42 genes restantes. En el siguiente paso (división del cluster C_1) del algoritmo, los 11 genes con menor varianza son separados del resto (cluster C_4). Estos genes presentaban un nivel constante de expresión. Algunos presentaban valores de su media de expresión altos (sobreexpresados) mientras otros presentaban valores bajos de expresión. El primer grupo está formado por genes activos mientras que en el otro grupo se encuentran los genes reprimidos. El grupo restante (cluster C_3) con 31 genes es de nuevo analizado, discriminando un grupo de 24 genes de menor varianza entre réplicas (cluster C_5), concluyendo que presentan una expresión bastante constante y por tanto importantes para

el fenotipo. El otro grupo (cluster C_6) formado por 7 genes con varianzas entre réplicas bastante elevadas y por tanto no son muy importantes para el carácter analizado.

Como muestra este ejemplo, este algoritmo da mucha importancia a la variabilidad entre réplicas. En cada partición, el algoritmo tiende a encontrar un cluster formado por genes con poca variabilidad entre réplicas y un segundo cluster con genes presentando gran variabilidad entre réplicas, indicando que son genes con nivel de expresión génica variable. Solamente más tarde el criterio relativo a la *path distance* empieza a tener importancia, por lo que el criterio relacionado con la variabilidad entre réplicas tiene prioridad.

Finalmente, este mismo procedimiento se ha probado también para otros dos conjuntos de datos. El primero corresponde al estudio de la clasificación de los melanomas cutáneos (Bittner *et al.*, 2000) y el segundo al enveje-cimiento en *C. elegans* (Lund *et al.*, 2002). No se presentan ambos ejemplos en detalle para no alargar innecesariamente la presente memoria. En ambos casos se obtuvieron también buenos resultados con el algoritmo desarrollado en este capítulo (ver para más detalles Irigoien *et al.* (2008)).

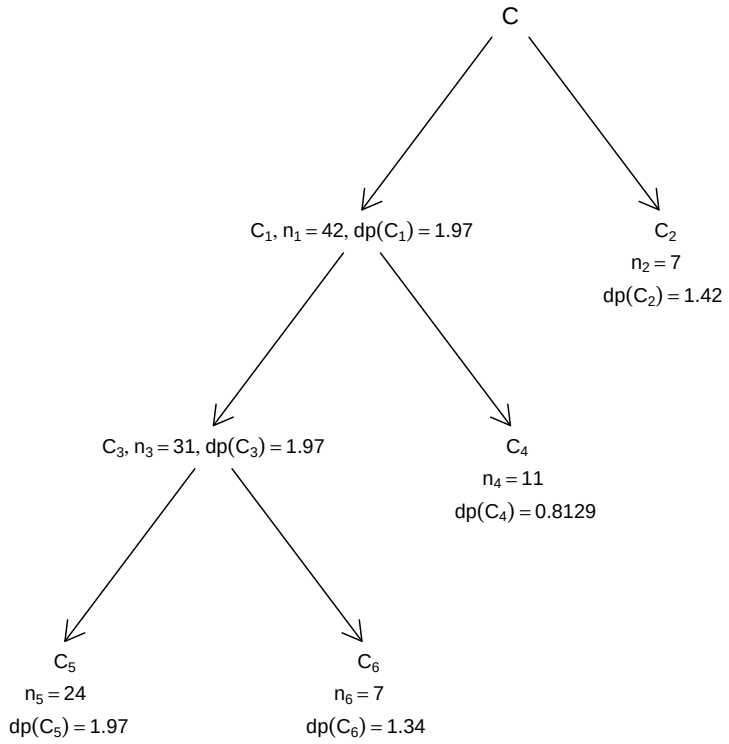


Figura 5.7: Representación gráfica de las particiones obtenidas sobre el conjunto de datos del estudio de los mecanismos funcionales del colesterol HDL. En cada paso se muestra el número de objetos de cada cluster y la correspondiente *path distance*

5.6. Conclusiones

En este capítulo se ha introducido un nuevo método de cluster divisivo basado en la que se ha denominado *path distance*. El método presentado no tiene límite respecto al número de objetos a clasificar, permite el uso de variables mixtas y es aplicable al caso de experimentos replicados.

El método soporta el análisis conjunto de los valores observados de las réplicas y de la variabilidad entre las mismas. Uno de los problemas de los métodos divisivos es el tiempo de ejecución, y es conocido que el tiempo de convergencia depende del algoritmo y de otros factores incluyendo la propia estructura de los datos. En simulaciones con más de 1000 objetos y utilizando un AMD Athlon(tm)XP 2000+, 1666MHz se han obtenido las clasificaciones de forma simple y rápida. La aplicación a conjuntos de objetos simulados y reales demuestra que se trata de un método válido para la clasificación de objetos y en particular, para los estudios de datos obtenidos de microarrays, detectando el posible comportamiento real de los genes.

5.7. Difusión del método

Una primera aproximación del algoritmo para el caso de una única variable continua y sin réplicas fue publicada en «Clum: A cluster program for analyzing microarray data», *Russian Journal of Genetics*, 44(4), 993-996, (2008). El caso de una única variable continua pero contemplando el caso replicado está publicado en

«Clapper: A clustering algorithm based on path-distances for experimental replicates», *Advances and Applications in Statistics*, 6, 379-400, (2006). Actualmente se esta preparando una futura publicación que contenga el caso general, presentado en este capítulo, de tener un conjunto de variables mixtas y considerando el caso replicado.

5.8. Software desarrollado

Para la aplicación del método presentado, se han desarrollado las siguientes funciones incluidas en el paquete `DBmethods`.

- `CLAPPER(d, renom, replicates = FALSE, DV = NULL, coef = 2/3)`: dada una matriz de distancias `d` correspondiente a los objetos de un conjunto C, calcula la división del conjunto C según la metodología presentada en este capítulo. El parámetro lógico `replicates = TRUE/FALSE` indicará en cada caso si se trata o no de un experimento con réplicas. Por defecto indica que no es un experimento con réplicas. Cuando el experimento es con réplicas, `DV` contiene las raíces cuadradas de las variabilidades geométricas entre las réplicas para cada objeto y el parámetro `coef` hace referencia al coeficiente A de manera que $\alpha = A(v_{max} - v_{min})$. Por defecto, $\alpha = 2/3(v_{max} - v_{min})$. El parámetro `renom` permite introducir identificadores de los objetos (Anexo A, pág. 219; Anexo B, pág. 260).

  Esta función necesita las funciones auxiliares `clapperSINreplicas`

(Anexo B, pág. 265), `buscar_solSINreplicas` (Anexo B, pág. 257), `clapperCONreplicas` (Anexo B, pág. 261), `buscar_sol` (Anexo B, pág. 254) y `condicion` (Anexo B, pág. 269).

Hay que señalar que esta función no desarrolla todo el proceso divisivo. Se ha construido en *matlab* un programa que sí realiza todas las sucesivas divisiones pero tal como se ha comentado anteriormente, por diferencias en el tratamiento de punteros que hace R, en esta versión del paquete `DBmethods` no se ha incluído una función que realice completamente el proceso divisivo. Está previsto construir en R una función que desarrolle todo el proceso divisivo.

- `dcor(x)`: dada una matriz de datos `x` calcula la matriz de distancias correlación descrita en el capítulo 2 (Anexo A, pág. 231; Anexo B, pág. 270).

- `path.d(d)`: dada una matriz de distancias `d` correspondiente a los objetos de un conjunto C, calcula la *path distance* del conjunto C, $dp(C)$ (Anexo A, pág. 239; Anexo B, pág. 303).

  Esta función necesita la función auxiliar `haysolucion` (Anexo B, pág. 294).

Capítulo 6

Patrones de expresión génica en el tiempo. Análisis de réplicas.

Este capítulo de la memoria está dedicado a la clasificación de los patrones de expresión génica en el tiempo, los denominados «time course experiments». Estos estudios de expresión se realizan tomando unas cuantas mediciones en pocos instantes temporales y es importante tener en cuenta que se realizan varias réplicas para cada experimento. Este tipo de experimentos ofrecen un enorme potencial para explorar los mecanismos subyacentes en diversos fenómenos biológicos y son empleados en muchas áreas de investigación biomédica (Slonim (2002), Storey *et al.* (2005)). Sin embargo, estos estudios tienen el inconveniente de generar una

gran cantidad de datos a tratar. Precisamente esto es lo que los hace interesantes desde el punto de vista del análisis de datos, ya que ofrecen nuevas situaciones y aparecen diversos retos a superar. Concretamente, tal como ya se ha comentado en la introducción de la memoria, en los denominados «time course experiments» con réplicas, el objetivo es clasificar un gran conjunto de genes en relación con su patrón de expresión teniendo siempre presente la información proporcionada por las réplicas. Por ello se define en el presente capítulo una nueva distancia que permita realizar de forma óptima dicha clasificación. Utilizando esta nueva distancia, el método que a continuación se presenta, filtra los genes con perfil plano, también identifica los genes con diferencias significativas entre réplicas y por último clasifica los genes en grupos según su patrón de expresión. La metodología que se desarrolla está inspirada en técnicas de análisis procrustes, pero debidamente modificadas para su correcta aplicación. A fin de evaluar la metodología propuesta, ésta se aplica a un conjunto de datos reales con una estructura ya conocida. Como es habitual en esta memoria, el capítulo finaliza con una descripción de las funciones desarrolladas para la fácil aplicación de las mismas.

6.1. Fundamentos teóricos

Consideremos pues que se ha realizado un experimento replicado con R réplicas, en el que se ha medido en nivel de expresión de G genes durante T instantes de tiempo (se supone que el número de instantes de observación es pequeño). En este tipo de experimentos, más que el valor concreto del instante de tiempo

en el que se mide el nivel de expresión interesa si es el primer, segundo, etc. punto de medición temporal, así se habla de punto t de medición en el tiempo, $t = 1, 2, \ldots, T$, siempre que el intervalo de tiempo transcurrido entre una medición y la siguiente sea el mismo. Esto no tiene porqué ser así ya que se pueden tener mediciones transcurridas, por ejemplo, 1, 2, 7, 9 y 10:30 horas, desde el incio del experimento. Sin embargo, es evidente que esta escala temporal sí debe ser la misma entre réplicas y entre genes.

Sea x_{grt} el nivel de expresión de un gen g, correspondiente a la r-ésima réplica y punto de medición t. Como en esta sección se trabaja con un gen cualquiera pero fijo, a fin de simplificar la notación se suprimirá en todo lo que sigue el subíndice g y todo hará referencia a dicho gen.

Sea $\mathbf{X}_r$ $(r = 1, \ldots, R)$ una matriz $T \times 2$ donde la primera columna contiene los puntos de medición en el tiempo $t_1, t_2, \ldots, t_T$ y la segunda columna contiene los valores de niveles de expresión $x_{rt_1}, x_{rt_2}, \ldots, x_{rt_T}$ del gen fijado. Por tanto fijado un gen, éste tiene asociadas R matrices del tipo anterior, una para cada una de las réplicas. En estas matrices, cada fila representa un punto de coordenadas (t_j, x_{rt_j}) respecto a ejes ortogonales en un espacio Euclídeo bidimensional. Es decir, estas matrices representan R configuraciones diferentes de puntos en el plano con (t_j, x_{rt_j}) como coordenadas, $j = 1, \ldots, T$, $r = 1, \ldots, R$.

El principal objetivo de este capítulo es comparar estas configuraciones $\mathbf{X}_1, \ldots, \mathbf{X}_R$ y medir el grado de asociación o similitud entre ellas. Para ello se utilizarán algunas ideas derivadas del análisis procrustes. Como ya se ha co-

mentado en la sección 2.2 del capítulo 2, el análisis procrustes se basa en tres procesos secuenciales que preservan en general la forma de un objeto: traslación, rotación/simetría y dilatación. Sin embargo, en el caso que nos ocupa no tiene ningún sentido considerar la operación rotación/simetría ya que ésta puede llevar a considerar como similares réplicas con perfiles muy diferentes. Por ejemplo, la Figura 6.1(a) muestra dos réplicas hipotéticas con perfiles claramente diferentes. La inclusión de una posible rotación (Figura 6.1(b)) asemejaría ambas réplicas y si se considerara como posible la realización de una simetría, ambas réplicas pasarían a ser totalmente coincidentes, tal como muestra la Figura 6.1(c). Por todo lo anterior, en el contexto que nos interesa se consideran los procedimientos de análisis procrustes pero omitiendo las operaciones de rotación/simetría. Así pues, a fin de estudiar la similitud de las configuraciones $\mathbf{X}_1, \ldots, \mathbf{X}_R$ los pasos serán los siguientes.

Inicialmente cada configuración es centrada, de forma que se considera su centro como origen de coordenadas. A continuación y con el fin de asegurar la conmensurabilidad entre las R configuraciones, cada matriz ya centrada se estandariza de manera que $\|\mathbf{X}_r - \bar{\mathbf{x}}_r\| = 1$, donde $\bar{\mathbf{x}}_r$ es el centro de la configuración correspondiente a la r-ésima réplica ($r = 1, \ldots, R$). Se recuerda que utilizaremos $\|\mathbf{A}\|$ para referirnos a la norma $tr(\mathbf{A}'\mathbf{A})$.

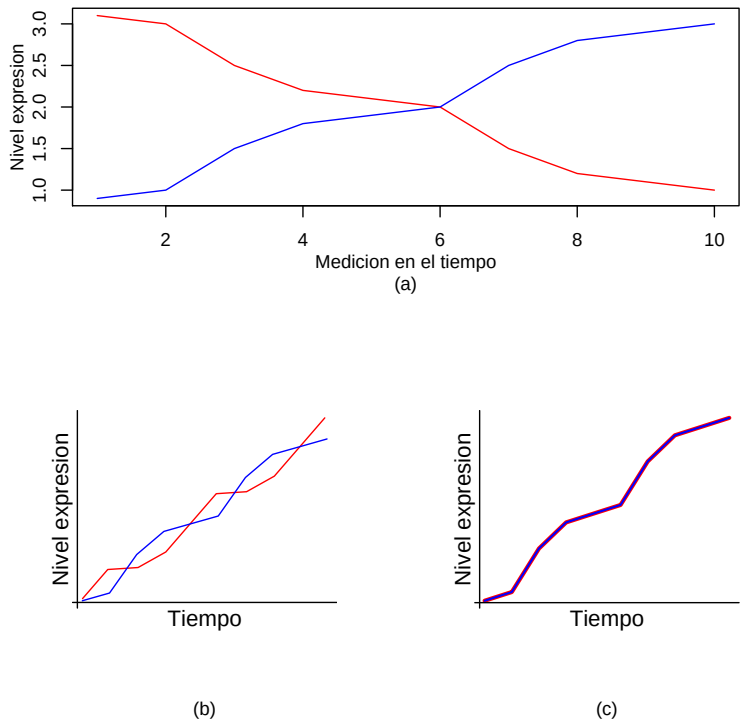


Figura 6.1: (a) Ejemplo hipotético de dos réplicas con perfiles diferentes. (b) Se observa que la inclusión de rotaciones haría que las réplicas fueran consideradas como más parecidas. (c) Finalmente, la inclusión de rotaciones generalizadas o simetrías implicaría la igualdad entre réplicas.

Supongamos que cada configuración $\mathbf{X}_r$ ha sido centrada y estandarizada, la última operación para medir el grado de similitud entre las configuraciones será la dilatación por factores s_r $(r = 1, \ldots, R)$, los cuales deben minimizar

$$\sum_{r<u} \|s_r \mathbf{X}_r - s_u \mathbf{X}_u\|. \tag{6.1}$$

A continuación todos los resultados y definiciones de análisis procrustes presentados de forma general en la sección 2.2 del capítulo 2, se reformulan teniendo en cuenta que se pretende evaluar la similitud entre las configuraciones $\mathbf{X}_1, \ldots, \mathbf{X}_R$ y que no se permite la operación de rotación /simetría.

El mínimo de la expresión (6.1) se alcanza tomando $\mathbf{s} = (s_1, \ldots, s_R)'$ como el vector propio asociado al mayor de los valores propios de la matriz

$$\mathbf{S} = \left(trace(\mathbf{X}_r' \mathbf{X}_u)\right)_{r,u=1,\ldots,R}.$$

Con el fin de evitar la solución degenerada de (6.1), se impone la restricción $\sum_r \|s_r \mathbf{X}_r\| = \sum_r \|\mathbf{X}_r\|$, que ahora se reduce a imponer $\mathbf{s}'\mathbf{s} = R$.

El valor mínimo de la expresión (6.1) es el estadístico procrustes asociado a las R réplicas y en todo lo que sigue se denotará por M_R.

En el nuevo contexto en el que se trabaja en este capítulo, la igualdad (2.16) pasa a ser:

$$\sum_{r<u} \|s_r \mathbf{X}_r - s_u \mathbf{X}_u\| = R \sum_{r=1}^{R} \|s_r \mathbf{X}_r - \mathbf{X}\mathbf{M}_R\|,$$

donde $\mathbf{XM}_R$ es la media procrustes ponderada definida como

$$\mathbf{XM}_R = \frac{1}{R}\sum_{r=1}^{R}(s_r\mathbf{X}_r),$$

y la descomposición de la variabilidad viene dada por la siguiente igualdad:

$$\sum_{r=1}^{R}\|s_r\mathbf{X}_r\| = \sum_{r=1}^{R}\|s_r\mathbf{X}_r - \mathbf{XM}_R\| - R\|\mathbf{XM}_R\|.$$

Hay que señalar que la media procrustes ponderada $\mathbf{XM}_R$ es la configuración que mejor se ajusta a las R réplicas del gen fijado.

En el caso particular de trabajar solamente con dos configuraciones ($R = 2$), el objetivo es determinar un único factor de escala s de forma que se minimice,

$$\|s\mathbf{X}_1 - \mathbf{X}_2\|.$$

Este mínimo se alcanza en el denominado estadístico procrustes M, que viene dado por

$$M = 1 - (trace\mathbf{X}_1'\mathbf{X}_2)^2 \tag{6.2}$$

y que se alcanza para $s = trace(\mathbf{X}_1'\mathbf{X}_2)$.

Finalmente, recordamos que el estadístico procrustes M toma valores entre 0 y 1. Valores cercanos a 0 indican una fuerte asociación entre las dos configuraciones, mientras que valores cercanos a 1 indican ausencia de asociación. Además Sibson (1978) probó que (6.2) define una métrica de manera que estos valores pueden ser tratados como distancias (al cuadrado) entre dos configuraciones.

Hay que hacer notar que, en nuestro contexto, el coeficiente de correlación, muy utilizado como medida de asociación, puede no ser una medida adecuada para estudiar la similaridad entre réplicas cuando el experimento tiene pocas mediciones temporales (Peddada *et al.*, 2003). En efecto, considérense por ejemplo dos réplicas hipotéticas como las presentadas en la Figura 6.2, que tienen perfiles bien diferenciados, donde un perfil crece monótonamente, y el otro muestra un pico en el cuarto punto de medición temporal. El coeficiente de correlación entre estas dos réplicas hipotéticas es $r = 0.907$, sugiriendo erróneamente que las dos réplicas muestran perfiles similares. En cambio, el valor $M = 0.369$ del estadístico procrustes indica correctamente que se trata de perfiles bien diferenciados. Por otra parte, la Figura 6.3 muestra dos hipotéticas réplicas con perfiles muy similares. Los dos perfiles muestran un pico en el cuarto punto de medición temporal, de manera que las diferencias no son biológicamente relevantes. En este caso, el coeficiente de correlación $r = 0.550$ sugiere perfiles diferentes, sin embargo el valor del estadístico procrustes $M = 0.044$ indica correctamente que los perfiles son similares. Obsérvese que como los dos perfiles son muy similares el coeficiente de dilatación es muy cercano a 1 ($s = 0.978$).

Por último, hay que hacer notar que los puntos de medición temporal $t = t_1, \ldots, t_T$ intervienen en los cálculos de los estadísticos procrustes por lo que no hay pérdida de información en cuanto al ordenamiento de los niveles de expresión a lo largo del tiempo.

Nota: En lo que sigue y con el fin de evitar posibles confusiones, $M_{g,R}$ denotará el valor del estadístico procrustes generalizado calculado utilizando las R

réplicas medidas sobre el gen g. De manera similar, $\mathbf{XM}_{g,R}$ denotará la media procrustes ponderada de las R réplicas medidas sobre el gen g.

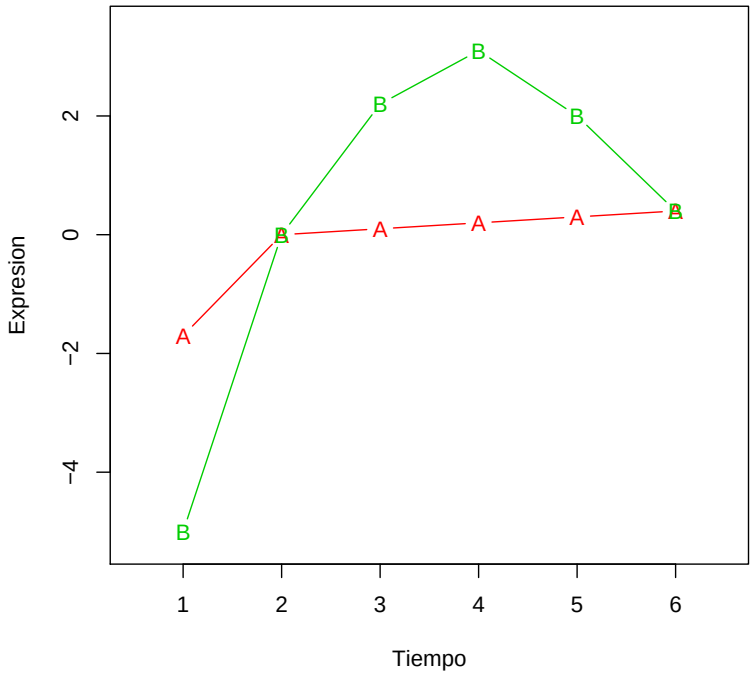


Figura 6.2: Ejemplo hipotético de dos réplicas con perfiles claramente diferentes. En esta situación el valor del estadístico procrustes $M = 0.369$ indica correctamente que los perfiles son bastante diferentes, mientras que el valor del coeficiente de correlación $r = 0.907$ indicaría erróneamente que son perfiles similares.

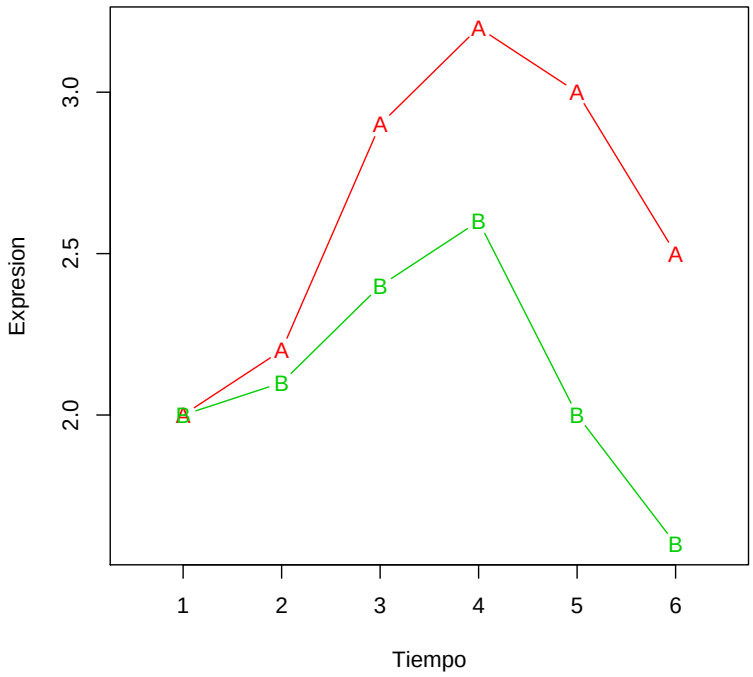


Figura 6.3: Ejemplo hipotético de dos réplicas con perfiles similares. En esta situación el valor del estadístico procrustes $M = 0.044$ indica correctamente que se trata de perfiles similares, mientras que el valor del coeficiente de correlación $r = 0.550$ indicaría, erróneamente, que son perfiles bastante diferentes.

6.2. Clasificación de patrones de expresión génica

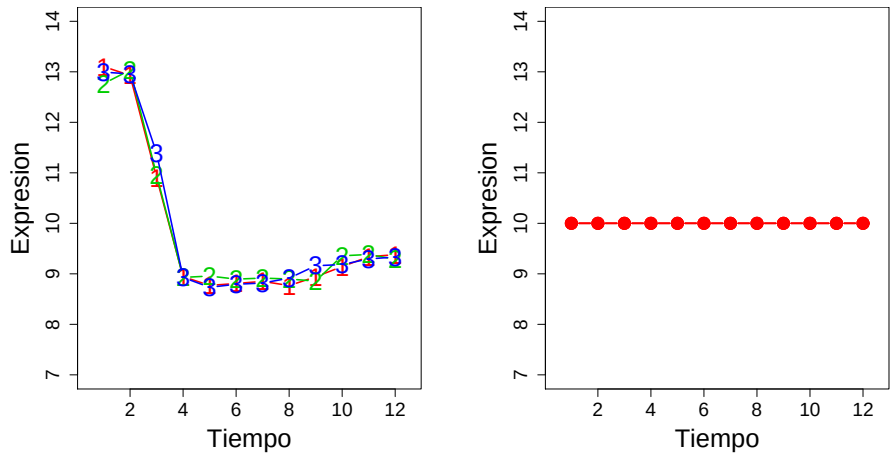
En esta sección se presenta toda la metodología necesaria a fin de realizar de forma correcta la clasificación de los patrones de expresión génica en un experimento replicado con R réplicas, en el que se ha medido el nivel de expresión de G genes durante T (con T pequeño) instantes de tiempo. La obtención de la clasificación final en un experimento de este tipo, requiere de toda una serie de pasos que se detallan a continuación.

PASO A. *Filtrado de genes.*

En este tipo de estudios el conjunto de genes que se analizan suele ser muy grande, y por tanto es conveniente filtrar dicho conjunto de genes. Concretamente, como muchos de los genes considerados en un principio no mostrarán cambios en su valor de expresión a lo largo de la duración del experimento, un primer paso será reducir el conjunto de genes filtrando aquellos genes que presenten un perfil de expresión plano o constante.

Definición 6.1. *Se define el* perfil nulo *como el perfil con iguales niveles de expresión en cada punto de medición* $t = t_1, \ldots, t_T$.

La Figura 6.4(a) muestra un ejemplo de un gen con tres réplicas prácticamente idénticas y presentando un importante cambio en los valores del nivel de expresión entre los tiempos dos y cuatro. Por su parte, la Figura 6.4(b) muestra un ejemplo de perfil nulo.



(a) (b)

Figura 6.4: (a) Gen con tres réplicas similares que muestran grandes cambios en su expresión. (b) Ejemplo de perfil nulo.

Obsérvese que, como para medir la similaridad entre réplicas a las configuraciones que las describen se les aplica en primer lugar una traslación, el valor constante del perfil nulo puede ser definido por el usuario de forma arbitraria. De hecho, lo más simple es tomar de entrada el valor constante igual a cero.

Para determinar aquellos genes cuyas réplicas sean muy diferentes al perfil nulo, para cada gen se evalúa el estadístico procrustes generalizado $M_{g,R+1}$ correspondiente a las R réplicas del gen g junto con el perfil nulo, es decir, se estarán comparando $R+1$ configuraciones. Una vez ordenados los valores de $M_{g,R+1}$ en orden decreciente, se separan los N genes que presentan valores más elevados. De hecho, fijado un gen g, cuanto mayor es el valor $M_{g,R+1}$, mayor es la diferencia entre el perfil de sus réplicas y el perfil nulo. El valor de N es fijado por el

investigador y de esta manera se separan los N genes con mayores cambios en los niveles de expresión a lo largo del experimento. De todas maneras, hay que tener en cuenta que si entre las réplicas de un mismo gen hay grandes diferencias, es decir, la variabilidad entre réplicas es muy grande, también se esperan valores altos de $M_{g,R+1}$. Por ello, en el siguiente paso del algoritmo se identifican los genes con diferencias importantes entre sus réplicas.

El método puede también aplicarse a un perfil nulo obtenido a través de los datos, es decir, a un perfil que varíe tan poco que sea claro que no existen diferencias de expresión entre los diferentes puntos de medición temporal. Algún perfil nulo puede ser seleccionado y utilizado en el algoritmo. De esta forma, la variabilidad en los perfiles no nulos se puede comparar con la variabilidad inherente de los datos, la cual viene representada por estos perfiles que no muestran variación de expresión entre tiempos de medición.

PASO B. *Estudio de la variabilidad entre réplicas.*

Tal como se acaba de comentar, podría ocurrir que valores elevados de $M_{g,R+1}$ fueran consecuencia de una gran variabilidad entre las réplicas del gen g. Por ello, es conveniente identificar estos genes ya que podría tratarse de genes especiales o bien que estas diferencias entre réplicas estuvieran indicando que se ha producido algún tipo de deficiencias en el proceso experimental. Por tanto, para cada gen g previamente seleccionado en el paso A, se considera el estadístico procrustes $M_{g,R}$ asociado a las R réplicas del gen. Si el valor de $M_{g,R}$ es muy pequeño indica que las réplicas son similares, y si no lo es, que son diferentes. Por tanto, genes con grandes valores de $M_{g,R}$ serían genes con gran variabilidad entre sus réplicas.

De todas maneras, si se desea mayor información acerca de la variabilidad entre las réplicas, proponemos el siguiente procedimiento de remuestreo bootstrap:

1. Para cada gen g y para cada punto de medición temporal t ($t = t_1, \ldots, t_T$), sea x_{grt} el valor del nivel de expresión en la r-ésima réplica ($r = 1, \ldots, R$). Para cada punto de medición temporal t, se selecciona con reemplazamiento uno de estos valores de expresión, es decir, un valor del conjunto $\{x_{g1t}, \ldots, x_{gRt}\}$. De esta forma se genera una nueva réplica, la $(R+1)$-ésima réplica.

2. Se calcula el estadístico procrustes generalizado M_{g,b^*} entre las R réplicas del gen incluyendo la nueva réplica generada en el paso 1. Se comparan los valores de $M_{g,R}$ con M_{g,b^*}, tomando $|M_{g,R} - M_{g,b^*}|$. La idea intuitiva subyacente a esta expresión es que valores cercanos a 0 indican que hay poca variabilidad entre las réplicas, mientras que valores altos indican gran variabilidad entre réplicas, puesto que en este caso la $R+1$-ésima réplica generada en el paso 1, sería una réplica con un perfil muy diferente a los determinados por las réplicas del gen.

3. Repitiendo B veces los pasos 1 y 2 se obtiene la distribución bootstrap de $|M_{g,R} - M_{g,b^*}|$ y por tanto se puede calcular su varianza. Dado un valor λ, se decide si la variabilidad entre las réplicas del gen g debe ser tenida en cuenta o no. Otra vez, será el investigador quien fije el valor λ después de consideraciones biológicas.

Como se ha comentado, es interesante preguntarse por qué hay genes con diferencias importantes entre sus réplicas, y por tanto para el investigador será útil

separar en dos grupos aquellos genes con poca y con gran variabilidad entre réplicas, a fin de hacer un estudio diferente entre los dos grupos. Así pues, en lo que sigue se denotará como N_{LV}, el conjunto de genes con poca variabilidad entre las réplicas y como N_{HL}, el conjunto de genes con gran variabilidad entre réplicas ($N = N_{LV} + N_{HV}$).

PASO C. *Distancia entre los genes de N_{LV}. Clasificación de los genes de N_{LV}.*

En este paso la atención se centra en los genes del grupo N_{LV}, es decir, entre todos aquellos genes que hemos seleccionado en el PASO A y que además no presentan diferencias importantes entre sus réplicas. A continuación se define un nuevo coeficiente de distancia entre dos genes de este grupo.

Definición 6.2.

Sean $\mathbf{XM}_g$ y $\mathbf{XM}_{g'}$ las medias procrustes ponderadas correspondientes a dos genes g y g' del subconjunto N_{LV}. Se define la distancia (al cuadrado) entre los dos genes $d^2_{g,g'}$, *como el estadístico procrustes entre $\mathbf{XM}_g$ y $\mathbf{XM}_{g'}$.*

La definición anterior, realmente es la definición de una distancia, ya que de acuerdo con Sibson (1978), la matriz cuadrada $\mathbf{D}^{(2)}$ cuyos elementos son los valores de $d^2_{g,g'}$ es una matriz de distancias (al cuadrado). Por tanto, una vez definida esta nueva distancia, para obtener una clasificación de los genes del grupo N_{LV} bastará con aplicar una técnica de cluster adecuada, como por ejemplo el método GEVA-Ward desarrollado en el capítulo 3 de esta memoria. Una vez obtenida una clasificación en clusters de los genes de N_{LV}, con el fin de determinar la calidad de la misma, se introducen los siguientes conceptos.

Definición 6.3.

Dado un cluster C_l, se define el perfil del cluster C_l *como la media procrustes ponderada obtenida utilizando todos los genes de C_l.*

Definición 6.4.

Se dice que un cluster C_l es compacto si todos sus genes tienen un perfil similar al perfil del cluster C_l.

Dadas las definiciones anteriores, y a fin de evaluar el grado de compacidad de un cluster C_l, se propone utilizar el siguiente estadístico,

$$T_l = \frac{1}{n_l} \sum_{g \in C_l} M_{g,C_l},$$

donde n_l representa el número de genes en el cluster C_l y M_{g,C_l} es el estadístico procrustes entre el gen g y el perfil del cluster C_l. Obsérvese que cuanto menor es el valor de T_l, más compacto es el cluster C_l. Tal y como es habitual, se puede obtener un intervalo de confianza bootstrap para cada T_l.

Por último, está claro que cuanto mayor sea el número de clusters k de una partición, mejor se ajustará cada gen al perfil de su cluster. Sin embargo, esto conlleva el riesgo de sobre-ajustar los datos. Como ya se ha comentado en capítulos anteriores, el tema de determinar el número de clusters es complejo y ya se ha propuesto una solución al mismo en el capítulo 4. En el contexto que ahora nos acupa, y dado que se tiene para cada cluster C_l el valor del estadístico T_l que expresa lo compacto que es el mismo, un modo sencillo de determinar el número de clusters utilizando esta información es calcular para cada posible

valor k, $k = 1, \ldots, K$, del número de clusters, el estadístico

$$PT_k = \frac{\sum_{l=1}^{k} n_l T_l}{\sum_l n_l}.$$

El menor de estos valores o bien el valor donde se estaciona PT_k, es el que nos proporcionará una estimación del número de clusters k en N_{LV}.

Hay que resaltar que en este paso del método se obtienen muchos resultados importantes. Por una parte se define una nueva distancia entre genes, la cual por todo lo comentado en la sección anterior no presenta los impedimentos o contradicciones que presentan otros coeficientes de asociación como el coeficiente de correlación. Por otra parte, la definición del perfil de un cluster proporciona el perfil representativo del cluster. Por último, hay que hacer notar que la determinación de estos perfiles de cluster no ha requerido en ningún momento tener prefijados posibles perfiles como candidatos a ser encontrados entre los genes en estudio.

PASO D. *Estudio de las diferencias entre réplicas. Clasificación de los genes de* N_{HV}.

Para un gen g del grupo N_{HV} hay dos cuestiones de interés: por una parte, identificar dónde están las diferencias entre sus réplicas, y por otra parte, decidir si el gen g pertenece a alguno de los clusters identificados en el paso C, o bien si tiene un perfil diferente.

Con el fin de identificar las diferencias entre las réplicas, fijado un gen g de N_{HV}, y fijadas dos cualesquiera de sus réplicas r y u $(r, u = 1, \ldots, R)$, se propone el test

$$H_0 : M_{g,(ru)} = 0, \tag{6.3}$$

donde $M_{g,2(ru)}$ denota el estadístico procrustes entre las r-ésima y u-ésima réplicas del gen g. La distribución del estadístico bajo H_0 se puede calcular mediante el siguiente método de remuestreo. Para empezar hay que decidir qué conjunto utilizar para obtener la distribución bajo H_0. Puesto que los genes del subconjunto N_{LV} cumplen la hipótesis nula, seleccionando B de estos mismos genes con reemplazamiento y calculando su correspondiente estadístico procrustes para las fijadas réplicas r y u, se obtiene la distribución bootstrap bajo H_0. Así, la hipótesis H_0 será rechazada si $M_{g,(ru)} \geq M^*_\alpha$, donde M^*_α es el percentil $(1-\alpha) \times 100$ de la distribución bootstrap. Se repite este procedimiento P veces para poder calcular el porcentaje de veces en las que la hipótesis H_0 ha sido rechazada. Este test permite identificar dónde están las diferencias entre las réplicas de un gen g. Puesto que este procedimiento se repite para cada gen y para cada par de réplicas, se debe considerar una corrección de tipo Bonferroni y por tanto, cuando el número de réplicas es alto, el test resulta muy conservador. Hay que comentar que existen otras técnicas adecuadas para este tipo de comparaciones múltiples. Así, el método False Discovery Rate (FDR) introducido por Benjamini and Hochberg (1995), es un método que se ha utilizado ampliamente en los métodos estadísticos de aplicación genómica. Para una buena revisión de los mismos, puede consultarse Storey and Tibshirani (2003) o bien Dudoit *et al.* (2003).

Por otra parte, para identificar si los genes de N_{HV} tienen perfiles no definidos en los clusters identificados en el paso C o si por el contrario estos genes sí que pertenecen a alguno de estos clusters, se propone utilizar el test INCA desarrollado en el capítulo 4 de esta memoria. Cabe mencionar que si las réplicas de un

gen g son muy diferentes, el perfil de la media procrustes ponderada puede no ser adecuada para representar al gen g y que por tanto habría que considerar el perfil de cada réplica por separado y hacer un estudio más en profundidad del comportamiento del gen.

6.3. Tiempos de ejecución

Puesto que en los «time course experiments» con réplicas la cantidad de información a manipular es muy grande, es importante tener en cuenta si un posible procedimiento propuesto para su estudio es o no realmente aplicable. Por ello queremos mencionar qué tiempos de ejecución cabe esperar de la metodología propuesta. Para los pasos A, B y C, la implementación en R sólo requiere unos pocos minutos de ejecución en un Intel Celeron M340 Processor, 1.5 GHz, 400 MHz FSB, para algo menos de 15000 genes, con un máximo de $R = 10$ réplicas y $T = 12$ mediciones temporales. El paso D consume más tiempo para la generación de la distribución nula del estadístico INCA (cerca de 6 horas en las condiciones que acabamos de mencionar para el número de genes y mediciones temporales con tres réplicas). Con estos tiempos de ejecución razonables, el método resulta viable.

6.4. Aplicación a datos reales

Con el objetivo de mostrar la bondad de esta metodología para identificar las diferencias entre réplicas y encontrar perfiles de clusters, se ha aplicado a un conjunto de datos sobre la embriogénesis de *Drosophila melanogaster*, del que se conocía su estructura de clusters y además se sospechaba que la segunda réplica presentaba para bastantes genes diferencias importantes respecto a las otras réplicas.

La embriogénesis en el organismo modelo *Drosophila melanogaster* ha sido profusamente estudiada (Wieschaus and Nüsslein-Volhard, 1998; Nüslein-Volhard, 2006; Moody, 2007). Los estudios de un gran número de genes han mostrado la diversidad en los patrones de expresión durante la embriogénesis de dicha especie y han puesto de manifiesto su importancia para el desarrollo de la expresión génica específica de tejido. Se ha podido observar que el cambio en los destinos celulares que tienen lugar durante su desarrollo va casi siempre asociado a cambios en la expresión génica. Por tanto, el conocimiento en detalle de los patrones espacial y temporal de las expresión de los genes implicados en este proceso sería indudablemente un paso importante para comprender las redes de regulación que rigen el desarrollo celular. En nuestro ejemplo se han usado los datos de 36 experimentos de hibridación mediante Gene Chip Drosophila Genome Array, utilizando el equipo y protocolos estándar de Affymetrix (Tomancak *et al.*, 2002). El principal objetivo era examinar la embriogénesis de *Drosophila* estudiando los patrones de expresión génica en el tiempo (*time-course experiments*). La

metodología experimental utilizada fue, de forma abreviada, la siguiente: moscas de la cepa Canton S se criaron en 12 cajas de poblaciones durante tres días con medio de cultivo suministrado cada 12 horas. A partir del cuarto día se recolectaban embriones cada hora durante doce horas consecutivas. El mismo procedimiento se repitió en tres días con lo que se obtuvieron tres réplicas. Posteriormente el RNA de las doce muestras de los embriones se extraía, se generaba el correspondiente cDNA (mediante el protocolo de amplificación y marcaje estándar de Affymetrix) y por último se hibridaba con el Gen Chip Drosophila Genome Array según el protocolo descrito en Lockhart *et al.* (1996). Por este procedimiento se examinaron 14010 genes. La base de datos que hemos utilizado para probar nuestro procedimiento estadístico contiene los valores de expresión génica hora a hora durante doce horas con tres réplicas. Por ello el procedimiento estadístico que se ha desarrollado en la sección anterior se ha aplicado a $G = 14010$ genes, con $R = 3$ réplicas y $T = 12$ puntos de medición temporal. La matriz de datos de los microarray que se ha usado está disponible en las páginas web: `http://www.fruitfly.org/cgi-bin/ex/insitu.pl` y `http://www.bioconductor.org`. A continuación se detallan los resultados que se obtuvieron.

PASO A: Para cada uno de los 14010 genes se calculó el valor del estadístico procrustes generalizado $M_{g,4}$ utilizando las tres réplicas del gen y el perfil nulo. Se consideró el 5 % de los genes con mayor valor de $M_{g,4}$ reduciéndose así el estudio a $N = 700$ genes. Por todo lo comentado con anterioridad, estos 700 genes pueden ser de dos tipos: genes con muy poca diferencia entre réplicas y por tanto con

grandes cambios en sus perfiles a lo largo del experimento, o bien, genes con gran variabilidad entre réplicas. Así pues se pasó a aplicar el siguiente paso.

PASO B: Para cada uno de estos 700 genes se calculó el valor de $M_{g,3}$, es decir, el valor del estadístico procrustes generalizado entre las tres réplicas del gen. Los valores de $M_{g,3}$ variaban de 0.003 a 0.210. Aplicando el procedimiento explicado de remuestreo, la varianza de $|M_{g,3} - M_{g,b^*}|$ variaba de 5.69×10^{-7} a 0.005. Tomando como valor $\lambda = 0.0001$, se separaron 629 genes como genes pertenecientes al grupo N_{LV} con valores de $M_{g,3}$ entre 0.003 y 0.042, y valores de la varianza de $|M_{g,3} - M_{g,b^*}|$ entre 5.69×10^{-7} y 9.63×10^{-5}. Los 71 genes restantes, el 10 % del total de 700 genes, se consideraron como elementos del grupo N_{HV}, con valores de $M_{g,3}$ entre 0.043 y 0.210, y valores de la varianza de $|M_{g,3} - M_{g,b^*}|$ entre 0.000102 y 0.005. En primer lugar se buscó una clasificación de los 629 genes del grupo N_{LV} siguiendo el siguiente paso.

PASO C: Para cada gen de N_{LV} se calculó el valor de la media procrustes ponderada, que como se ha mencionado a lo largo del capítulo es la configuración que mejor se aproxima a todas las réplicas del gen. Para cada par de genes se buscó el valor de la distancia entre ambos según la definición 6.2, obteniéndose una matriz de distancias de orden 629×629. Como método de cluster se consideró el método GEVA-Ward que se definió en el capítulo 3, y que se comprobó que proporcionaba buenos resultados en la clasificación de los genes. A fin de determinar el número de clusters, se evaluó el estadítico PT_k variando $k = 2, \ldots, 50$, observándose que dichos valores se estacionaban para $k = 15$. Así pues, se con-

sideraron 15 clusters que se identificaron por C_l $(l = 1, \ldots, 15)$ de forma que se siguió en la numeración de los clusters el orden de aparición en el correspondiente dendrograma.

La Tabla 6.1 muestra el tamaño de cada cluster C_l, el valor del estadístico T_l y su correspondiente intervalo de confianza al 95 %, obtenido a partir de 1000 sub-muestras bootstrap. Como es fácil observar, los resultados que proporciona el método GEVA-Ward son altamente satisfactorios, ya que los 15 clusters obtenidos son muy compactos.

Gracias a la metodología desarrollada, se puede asociar a cada cluster un perfil, el denominado perfil de cluster (definición 6.3), representativo de los perfiles de todos los genes que forman parte de dicho cluster. Estos 15 perfiles se muestran en las Figuras 6.7 - 6.21 que se presentan en un anexo al final del capítulo (pág. 190). Hay que comentar que estos 15 perfiles dan lugar a 7 perfiles generales: descendente-constante, constante-ascendente, ascendente-constante, monótono decreciente, monótono creciente, descendente-ascendente y ascendente-decreciente. Pero es importante hacer notar que la metodología desarrollada es capaz de distinguir perfiles más concretos, detallando más la estructura de los genes y mostrando perfiles no tan generales como los arriba mencionados. Así se observa que el perfil descendente-constante que se muestra en los clusters C_1, C_2, y C_5 (Figuras 6.7, 6.8 y 6.9) no es exactamente igual en los tres clusters. Mientras que en C_2 cae rápidamente después de la tercera medición temporal, en los otros dos el perfil cae rápidamente después del segundo tiempo de medi-

Tabla 6.1: Resultados obtenidos al aplicar la metodología desarrollada a los datos de embriogénesis en el organismo modelo *D. melanogaster*. Las columnas de la tabla muestran para los 15 clusters que se han obtenido: el tamaño del cluster, el valor del estadístico T_l y el correspondiente intervalo de confianza al 95 % obtenido a partir de 1000 sub-muestras bootstrap.

Cluster	*Tamaño cluster*	T_l	*I. C. bootstrap*
C_1	64	0.00951	(0.00767 , 0.01150)
C_2	54	0.00914	(0.00759 , 0.01089)
C_3	31	0.01260	(0.00881 , 0.01750)
C_4	13	0.01697	(0.01334 , 0.02099)
C_5	42	0.01476	(0.01153 , 0.01801)
C_6	60	0.01394	(0.01126 , 0.01696)
C_7	11	0.00955	(0.00679 , 0.01317)
C_8	38	0.02253	(0.01632 , 0.03051)
C_9	30	0.00814	(0.00615 , 0.01010)
C_{10}	62	0.00875	(0.00631 , 0.01168)
C_{11}	25	0.00895	(0.00715 , 0.01120)
C_{12}	10	0.00569	(0.00362 , 0.00834)
C_{13}	64	0.00895	(0.00715 , 0.01120)
C_{14}	54	0.01468	(0.01223 , 0.01758)
C_{15}	24	0.01844	(0.01527 , 0.02165)

ción, en especial en C_5. El perfil constante-ascendente lo presentan los clusters C_{10} (Figura 6.10) y C_{12} (Figura 6.11), creciendo después del noveno y octavo punto de medición temporal respectivamente. Los perfiles, ascendente-constante

y monótono decreciente sólo aparecen en los clusters C_9 y C_8, respectivamente (Figuras 6.12 y 6.13). Por otra parte, los clusters C_3, C_7 y C_{13} (Figuras 6.14, 6.15 y 6.16) siguen el perfil monótono creciente pero sus formas son bien diferentes. El perfil descendiente-ascendente se da en los clusters C_4 y C_{15} (Figuras 6.17 y 6.18), pero solamente el cluster C_{15} presenta un único y clarísimo mínimo en el cuarto punto de medición temporal. Finalmente, en los clusters C_6, C_{11} y C_{14} (Figuras 6.19, 6.20 y 6.21) aparece el perfil ascendente-decreciente, con máximos claros en el punto de medición temporal 3 para el cluster C_{14} y en el cuarto punto de medición temporal para el cluster C_{11}.

Aunque no es objetivo de esta memoria, es posible analizar qué significado biológico tienen estos 15 clusters. Por ejemplo, para el cluster C_1 se han examinado las funciones descritas para todos sus genes y se ha buscado si algún tipo de funcionalidad era predominante. Las funciones de los genes y los procesos biológicos en que éstos están involucrados se obtuvieron a partir del Gene Ontology Consortium (2000) mediante la aplicación *GeneMerge Web-tool* (Castillo-Davis and Hartl, 2003). El cluster C_1 (valor del estadístico = 0.0095) presentaba una representación importante de genes involucrados en la regulación de la actividad GTPasa (raw - score = 0.0001, e - score = 0.0056) y en el transporte mediado por vesículas (raw- score = 0.0011, e - score = 0.0377). Además, los clusters obtenidos se pueden asociar a tejidos concretos. Así por ejemplo, todos los genes del cluster C_9 se expresan en la epidermis dorsal de la cabeza en el primordio P1 y ningún otro gen de cualquiera de los otros clusters se expresa en este tejido.

PASO D: El último paso del método propuesto, centra su atención en los 71

genes de N_{HV}, es decir, aquellos genes que se han separado por presentar diferencias que se han considerado importantes entre sus réplicas (los valores de la varianza entre réplicas variaban de 0.000102 a 0.005). A fin de determinar la distribución bajo la hipótesis nula (6.3) del estadístico $M_{g,(ru)}$ se tomaron los valores $B = 1000$ y $P = 100$. El test no detectó diferencias realmente significativas entre 16 del total de los 71 genes. Únicamente dos genes (3.6 %) presentaron diferencias significativas entre las réplicas 1 y 3, mientras que los genes restantes mostraron diferencias significativas entre las réplicas 1 y 2 (9.4 %), o bien entre las réplicas 2 y 3 (87 %). Este hecho confirmaba, tal como se intuía, que los pequeños cambios que se produjeron durante la segunda réplica influyeron de forma importante en el perfil de algunos de los genes. Se observó que 38 genes mostraron un nivel de expresión especial en el décimo punto de medición temporal. Concretamente y tal como se muestra en la Figura 6.5, en el punto 10, el nivel de expresión de la segunda réplica decae para luego volver a recuperarse en el onceavo punto de medición temporal.

Finalmente se aplicó el test INCA (ver capítulo 4) con el fin de clasificar estos genes de N_{HV} en alguno de estos 15 clusters que tenemos identificados o bien para determinar si definían nuevos perfiles. La aplicación de dicho test identificó únicamente 3 genes como atípicos y que por tanto definían un nuevo perfil diferente de los 15 ya identificados. Los perfiles de estos 3 genes se muestran en la Figura 6.6. Obsérvese que para los genes *141541_at* y *146674_at* aunque las tres réplicas sean significativamente diferentes, las tres presentan un perfil similar y por tanto las correspondientes medias procrustes ponderadas proporcionan el perfil de dos

nuevos clusters C_{16} y C_{17} (Figuras 6.22 y 6.23 del anexo respectivamente, pág. 198). Como es fácil observar, estos dos nuevos perfiles presentan diferencias, pero se podrían agrupar en un perfil ascendente-decreciente-ascendente más general. Sin embargo, el gen *AFFX-Dros-18S_rRNA_M_at* presenta enormes diferencias entre sus tres réplicas, no pareciendo posible deducir su perfil.

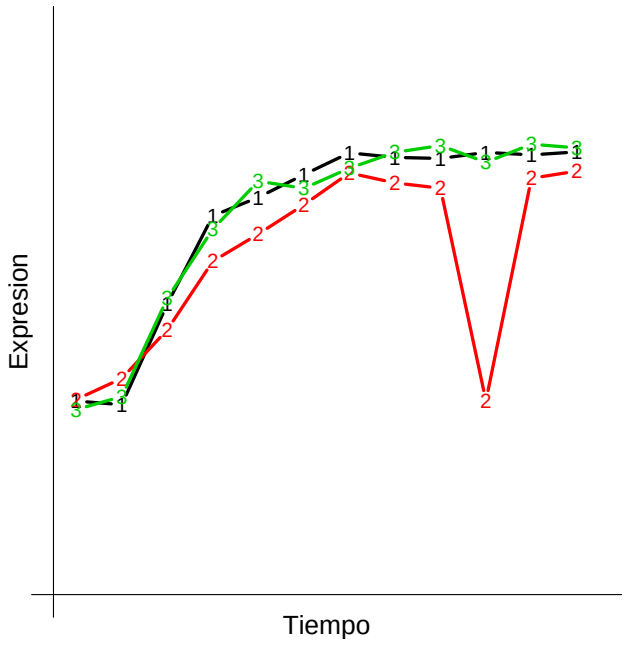


Figura 6.5: Se muestra el perfil de un gen del grupo N_{HV} con diferencias significativas entre sus réplicas. La segunda réplica decae en el décimo punto de medición temporal, recuperándose en el decimoprimero. Este comportamiento se detectó en 38 de los 71 genes de N_{HV}.

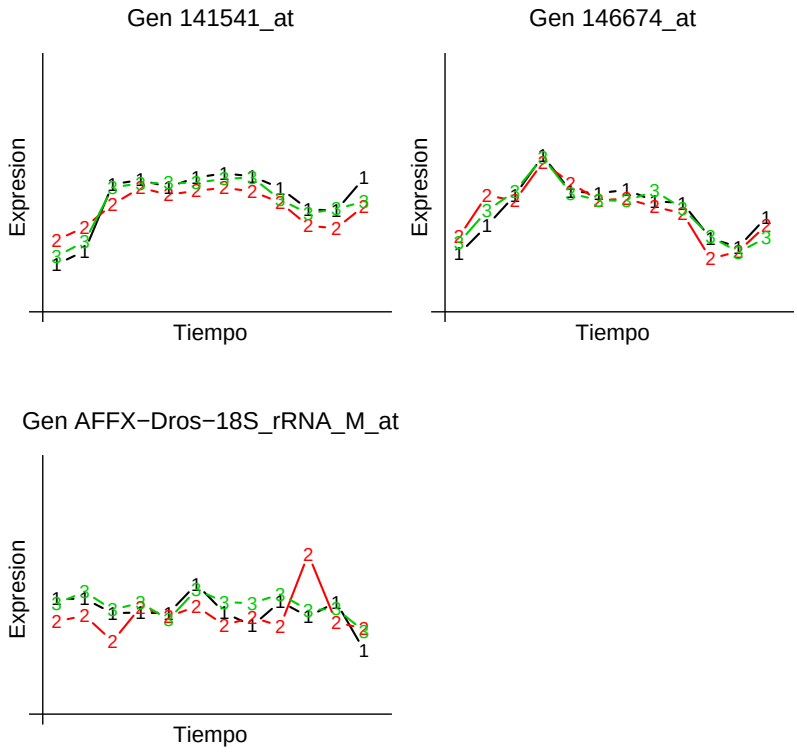


Figura 6.6: Genes del conjunto N_{HV} que se han identificado como atípicos, es decir, estos tres genes definen perfiles diferentes a los 15 identificados con anterioridad.

6.5. Conclusiones

La mayor contribución de este capítulo radica en la presentación de una nueva media entre las réplicas de un gen y una nueva distancia entre genes, basándose únicamente en la forma del perfil de expresión del gen. Este procedimiento selecciona los genes de expresión significativamente diferente del perfil nulo, identifica los genes sin diferencias significativas entre réplicas y los clasifica en clusters de perfil similar. El procedimiento ofrece un criterio para determinar el número de clusters y se clasifican los genes con diferencias entre sus réplicas en los clusters previamente identificados o bien son identificados como genes con nuevo perfil.

Este procedimiento difiere de otros en que no necesita perfiles previos, ni distribuciones de probabilidad ni otro tipo de restricciones. Además, este procedimiento puede detectar genes con diferencias entre réplicas. Por tanto, el procedimiento es menos restrictivo y más general que los procedimientos basados en el coeficiente de correlación. Además, este procedimiento ofrece un perfil medio del cluster con la media procrustes ponderada así como una medida de la compacidad del cluster.

La aplicación de nuestro procedimiento a los datos del experimento temporal de microarray de la embriogénesis en el organismo modelo *Drosophila melanogaster* proporciona resultados satisfactorios. El procedimiento ha identificado 7 perfiles generales previamente descritos por otros métodos, la mayoría de los cuales tienen significado biológico. Es más, nuestro procedimiento distingue subperfiles que pueden ser de interés. El procedimiento detecta diferencias en

la segunda réplica e identifica genes con un nivel de expresión particular en el décimo punto de medición de la segunda réplica. Puesto que el procedimiento es un método general, puede ser aplicado a otro tipo de estudios temporales en los que se pretenda encontrar diferentes formas funcionales.

6.6. Difusión del método

Los resultados que se muestran en este capítulo fueron presentados ante la comunidad científica en el *First Workshop of the ERCIM Working Group on Computing and Statistics*, 2008, celebrado en Neuchâtel (Suiza). Además estos resultados forman parte del artículo «Microarray time course experiments: finding profiles», que actualmente está sometido para su publicación.

6.7. Software desarrollado

El paquete `DBmethods`, además de las funciones señaladas en los capítulos 3, 4 y 5, contiene las siguientes funciones para poder aplicar el método desarrollado en este capítulo:

- `proc2(X, Y)`: dadas las expresiones temporales `X` e `Y` de dos perfiles calcula el estadístico procrustes M asociado (Anexo A, pág. 241; Anexo B, pág. 305).

  Esta función necesita de la función auxiliar `center` (Anexo B, pág. 259).

- `prock(dat)`: dadas las expresiones temporales de R perfiles ordenadas por filas en una matriz `dat`, calcula el estadístico procrustes generalizado M_R, así como la configuración media procrustes ponderada y los correspondientes coeficientes de dilatación (Anexo A, pág. 243; Anexo B, pág. 306).

  Esta función necesita la función auxiliar `center`.

- `serieprock(dat, nrep, g, null.p = FALSE)`: dadas las R expresiones temporales para un total de G genes ordenadas por filas en una matriz `dat`, indicando el número de réplicas R en el parámetro `nrep`, calcula los estadísticos procrustes generalizados M_R para cada gen y los devuelve ordenados de mayor a menor. El parámetro lógico `null.p = TRUE/FALSE` permite incluir o no el perfil nulo junto con los R perfiles dados para cada gen. Por defecto, no se incluye el perfil nulo. Además el parámetro `g` permite introducir identificadores para los genes (Anexo A, pág. 247; Anexo B, pág. 311).

  Esta función necesita las funciones auxiliares `center` y `prock_array` (Anexo B, pág. 308).

- `varepli(dat, nrep, g, np)`: dadas las R expresiones temporales para un total de G genes ordenadas por filas en una matriz `dat` e indicando el número de réplicas R en el parámetro `nrep`, calcula la variabilidad entre las réplicas de un mismo gen para cada gen tal y como se explica en el paso B de la sección 6.2 y los devuelve ordenados de mayor a menor.

  El parámetro `np` recoge el tamaño de la muestra bootstrap que se generará y el parámetro `g` permite incluir identificadores para los genes (Anexo A,

pág. 249; Anexo B, pág. 314). Esta función necesita las funciones auxiliares `center`, `prock_array` y `varb` (Anexo B, pág. 313).

6.8. ANEXO:

Gráficos de los perfiles de los clusters $C_1, \ldots, C_{15}$

A continuación se adjuntan las Figuras correspondientes a los 15 perfiles de los clusters determinados en el PASO C del procedimiento desarrollado en este capítulo. Además se incluyen los perfiles de dos nuevos clusters formados por los genes identificados como atípicos en el PASO D.

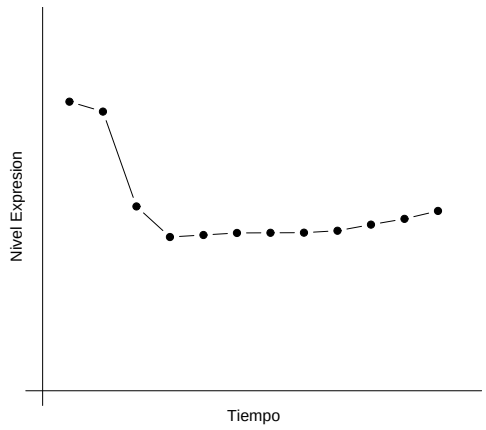


Figura 6.7: Perfil del cluster C_1. Perfil decreciente con gran salto entre los puntos de medición 2 y 3 y prácticamente constante a continuación.

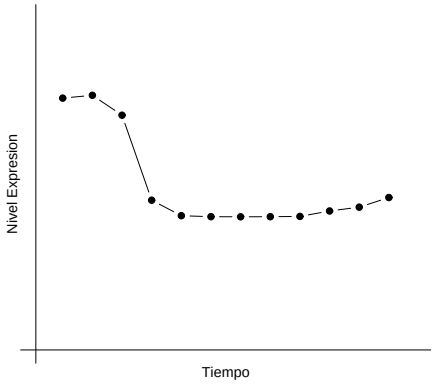


Figura 6.8: Perfil del cluster C_2. Perfil decreciente con gran salto entre los puntos de medición 3 y 4 y prácticamente constante a continuación.

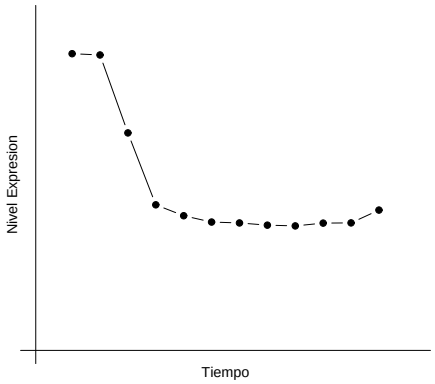


Figura 6.9: Perfil del cluster C_5. Perfil decreciente con gran salto entre los puntos de medición 2 a 4 y prácticamente constante a continuación.

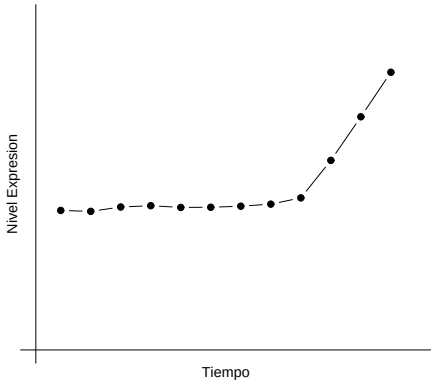


Figura 6.10: Perfil del cluster C_{10}. Perfil creciente práctimente constante hasta el noveno punto de medición y con gran salto a continuación.

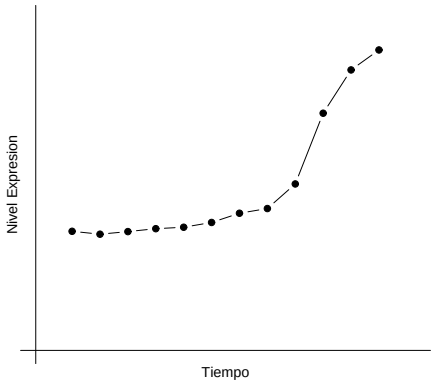


Figura 6.11: Perfil del cluster C_{12}. Perfil creciente práctimente constante hasta el octavo punto de medición y con gran salto a continuación.

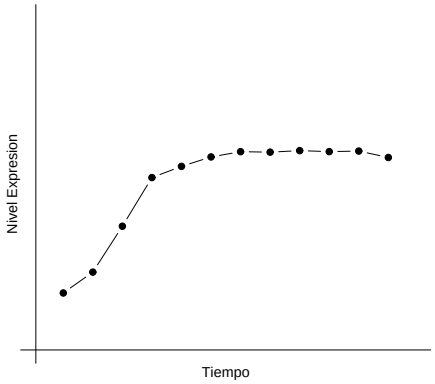


Figura 6.12: Perfil del cluster C_9. Perfil creciente con gran salto entre los puntos de medición 2 y 4 y prácticamente constante a continuación.

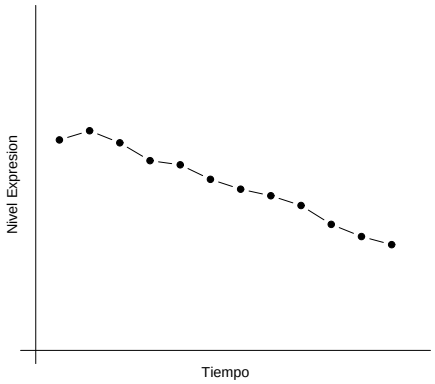


Figura 6.13: Perfil del cluster C_8. Perfil decreciente casi monótono.

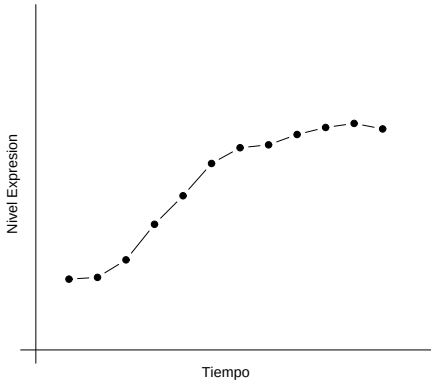


Figura 6.14: Perfil del cluster C_3. Perfil creciente monótono.

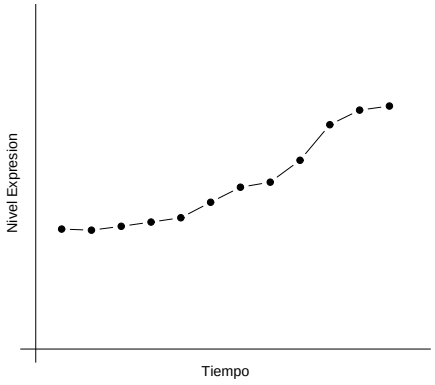


Figura 6.15: Perfil del cluster C_7. Perfil creciente casi monótono.

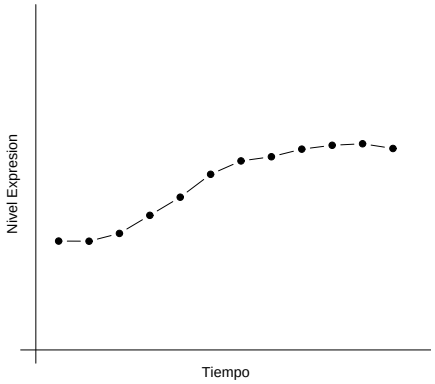


Figura 6.16: Perfil del cluster C_{13}. Perfil creciente monótono.

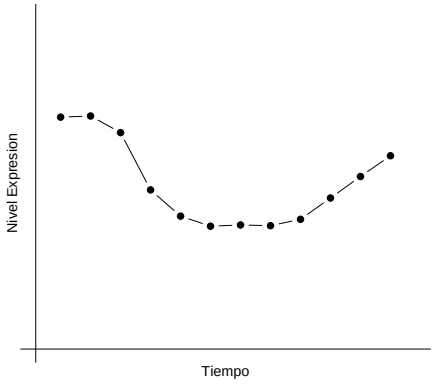


Figura 6.17: Perfil del cluster C_4. Perfil descendente-ascendente con mínimo en los puntos del medición 6, 7 y 8.

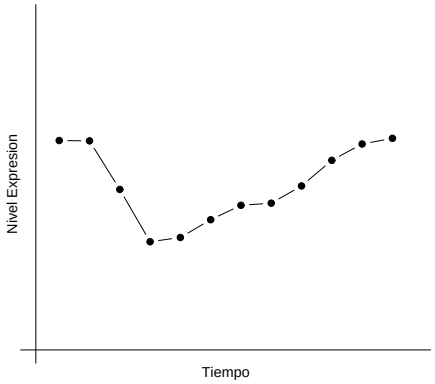


Figura 6.18: Perfil del cluster C_{15}. Perfil descendente-ascendente con mínimo en el cuarto punto de medición.

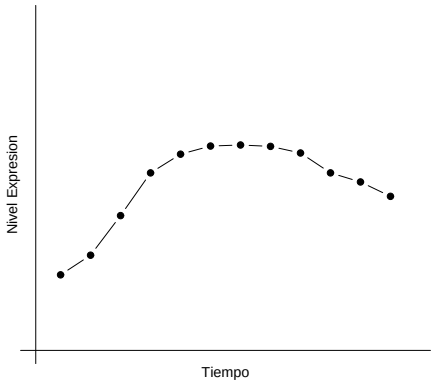


Figura 6.19: Perfil del cluster C_6. Perfil ascendente-descendente con máximo en los puntos de medición 6, 7 y 8.

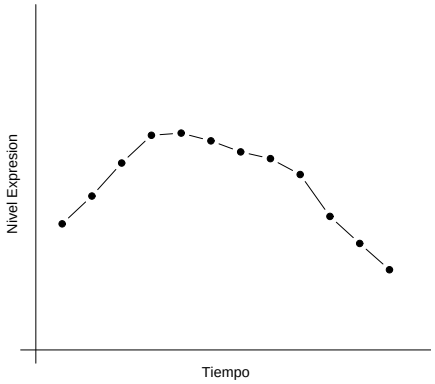


Figura 6.20: Perfil del cluster C_{11}. Perfil ascendente-descendente con máximo en los puntos de medición 4 y 5.

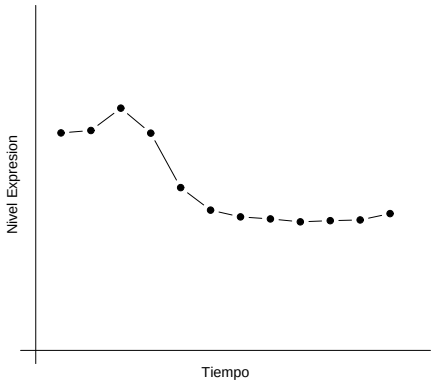


Figura 6.21: Perfil del cluster C_{14}. Perfil ascendente-descendente con máximo en el tercer punto del medición.

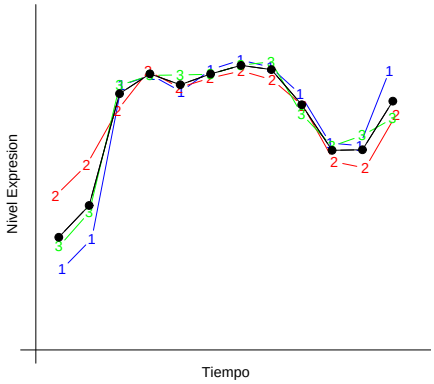


Figura 6.22: Perfil del cluster C_{16}. Perfil obtenido a partir del gen *141541_at*. También se muestran los perfiles de las tres réplicas de dicho gen.

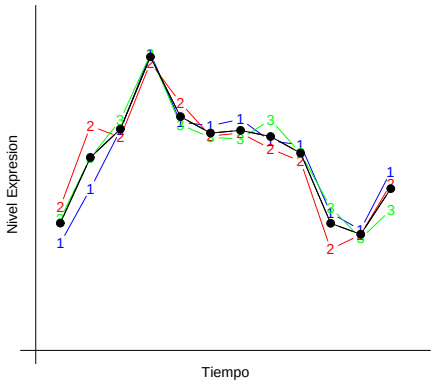


Figura 6.23: Perfil del cluster C_{17}. Perfil obtenido a partir del gen *146674_at*. También se muestran los perfiles de las tres réplicas de dicho gen.

Bibliografía

Anderson, M. J. and Robinson, J. (2003). Generalized discriminant analysis based on distances. *Australian and New Zealand Journal of Statistics*, **45**, 301–318.

Anderson, M. J. and Willis, T. J. (2003). Canonical analysis of principal coordinates: A useful method of constrained ordination for ecology. *Ecology*, **84**, 511–525.

Arenas, C. and Cuadras, C. M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

Ayala, F. J., Serra, L. and Prevosti, A. (1989). A grand experiment in evolution: the *D. subobscura* colonization of the Americas. *Genome*, **31**, 246–255.

Balanyà, J., Solé, E., Oller, J. M., Sperlich, D. and Serra, L. (2004). Long-term changes in chromosomal inversion polymorphism of *D. subobscura*. II. European populations. *Journal of Zoological Systematics and Evolutionary Research*, **42**, 191–201.

Bar-Hen, A. (2001). Preliminary tests in linear discriminat analysis. *Statistica*, **4**, 585–593.

Bar-Joseph, Z., Gerber, G., Jaakkola, T. S., Gifford, D. K. and Simon, I. (2003). Continuous representations of time series gene expression data. *Journal of Computational Biology*, **3–4**, 341–356.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the False Discovery Rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, **57**, 289–300.

Bennett, J. and Dutcher, T. (1969). The cytochemistry of acute leukemia: Observations on glycogen and neutral fat in bone marrow aspirates. *Blood*, **33**, 341–347.

Bhattacharyya, A. (1946). On a measure of divergence of two multinominal populations. *Sankhyā. The Indian Journal of Statistics*, **7**, 401–406.

Biau, G., Cadre, B. and Pelletier, B. (2007). A graph-based estimator of the number of clusters. *ESAIM: Probability and Statistics*, **11**, 272–280.

Bittner, M., Meltzer, P., Chen, Y., Jiang, Y., Seftor, E. , Hendrix, M., Radmacher, M., Simon, R., Yakhini, Z., Ben-Dor, A., Sampas, N., Dougherty, E., Wang, E., Marincola, F., Gooden, C., Lueders, J., Glatfelter, A., Pollock, P., Carpten, J., Gillanders, E., Leja, D., Dietrich, K., Beaudry, C., Berens, M., Alberts, D., Sondak, V., Hayward, N., and Trent, J. (2000). Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, **406**, 536–540.

Calinski, R. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, **3**, 1–27.

Callow, M. J., Dudoit, S., Gong, E. L., Speed, T. P. and Rubin, E. M. (2000). Microarray expression profiling identifies genes with altered expression in HDL deficient mice. *Genome Research*, **10**, 2022–2029.

Castillo-Davis, C. I. and Hartl, D. L. (2003). GeneMerge-post-genomic analysis, data mining and hypothesis testing. *Bioinformatics*, **19**, 891–892.

Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. O. and Herskowitz, I. (1998). The transcriptional program of sporulation in budding yeast. *Science*, **282**, 699–705.

Clark, P. and Niblett, T. (1987). Induction in noisy domains. In: *Progress in Machine Learning (from the Proceedings of the Second European Working Session on Learning)*, pp. 11–30. Sigma Press, UK.

Cliff, N. (1966). Orthogonal rotation to congruence. *Psychometrika*, **31**, 33–42.

Consortium, I. H. G. S. (2001). Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Consortium, I. H. G. S. (2004). Finishing the euchromatic sequence of the human genome. *Nature*, **431**, 931–945.

Cuadras, C. M. (1992). Some examples of distance based discrimination. *Biometrical Letters*, **29**, 3–20.

Cuadras, C. M. and Arenas, C. (1990). A distance based regression model for prediction with mixed data. *Communications in Statistics A. Theory and Methods*, **19**, 2261–2279.

Cuadras, C. M. and Fortiana, J. (1995). A continuous metric scaling solution for a random variable. *Journal of Multivariate Analysis*, **32**, 1–14.

Cuadras, C. M. and Fortiana, J. (2000). The importance of geometry in multivariate analysis and some applications. In: *Statistics for the 21st Century*, pp. 93–108. Marcel Dekker, New York.

Cuadras, C. M., Arenas, C. and Fortiana, J. (1996). Some computational aspects of distance-based model for prediction. *Communications in Statistics. Simulation and Computation*, **25**, 593–609.

Cuadras, C. M., Fortiana, J. and Oliva, F. (1997). The proximity of an individual to a population with applications in discriminant analysis. *Journal of Classification*, **14**, 117–136.

Dudoit, S. and Fridlyand, J. (2002). A prediction-based resampling method for estimating the number of clusters in a data set. *Genome Biology*, **3**, research0036.1–0036.21.

Dudoit, S., Shaffer, J. P. and Boldrick, J. C. (2003). Multiple hypothesis testing in microarray experiments. *Statistical Science*, **18**, 71–103.

Edwards, A. W. F. and Cavalli-Sforza, L. L. (1965). A method for cluster analysis. *Biometrics*, **21**, 362–375.

Eisen, M. B., Spellman, P., Brown, P. O. and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, **95**, 14863–14868.

Ernst, J., Nau, G. and Bar-Joseph, Z. (2005). Clustering short time series gene expression data. *Bioinformatics*, **21**, i159–i168.

Everitt, B. S. (1993). *Cluster analysis*, 3rd edition. John Wiley and Sons, New York.

Ferrandiz, C. (1996). *Dermatosis eritematoescamosas (I). Psoriasis. Eritrodemias. Dermatología clínica*. MMI Elsevier España, Madrid.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, 179–188.

Fowlkes, E. B. and Mallows, C. L. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, **78**, 553–584.

Fraley, C. and Raftery, A. (2002). Model-based clustering, discriminat analysis, and density estimation. *The Computer Journal*, **97**, 611–631.

Frei, E., Freireich, E., Gehan, E., Pinkel, D., Holland, J., Selawry, O., Haurani, F., Spurr, C. L., Hayes, D. M., James, G. W., Rothberg, H., Sodee, D. B., Rundles, R. W., Schroeder, L. R., Hoogstraten, B., Wolman, I. J., Traggis, D. G., Cooper, T., Ebaugh, F., Taylor, R. (1961). Studies of sequential and combination antimetabolite therapy in acute leukemia: 6-mercaptopurine and methotrexate. *Blood*, **18**, 431–454.

The Gene Ontology Consortium (2000). Gene Ontology: Tool for the unification of biology. *Nature Genetics*, **25**, 25–29.

Golub, T. R., Barker, G., Bohlander, S., Hiebert, S., Ward, D., Bray-Ward, P., Morgan, E., Raimondi, S. C., Rowley, J. D. and Gilliland, D. G. (1995). Fusion

of the *tel* gene on 12p13 to the *aml1* gene on 21q22 in acute lymphoblastic leukemia. *Proceedings of the National Academy of Sciences of the United States of America*, **92**, 4917–4921.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. and Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.

Goodall, C. (1991). Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society: Series B*, **53**, 285–339.

Gordon, A. D. (1999). *Classification*, 2nd edition. Monograph on Statistics and Applied Probability. Chapman and Hall, Boca Raton, USA.

Gower, J. C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, **53**, 325–338.

Gower, J. C. (1971a). A general coefficient of similarity and some of its properties. *Biometrics*, **27**, 857–871.

Gower, J. C. (1971b). Statistical methods for comparing different multivariate analysis of the same data. In: *Mathematics in the Archaeological and Historical Sciences*, pp. 138–149. Hodson, J. R., Kendall, D. G. and Tautu, P. (eds.). Edinburgh University Press, Edinburgh.

Gower, J. C. (1975). Generalized procrustes analysis. *Psychometrika*, **40**, 33–51.

Gower, J. C. (1985). Measures of similarity, dissimilarity and distance. In: *Ency-*

clopedia of Statistical Sciences, volume 5, pp. 397–405. John Wiley and Sons, New York.

Gower, J. C. and Dijksterhuis, G. (2004). *Procrustes Problems*. Oxford University Press, New York.

Gower, J. C. and Krzanowski, W. J. (1999). Analysis of distance for structured multivariate data and extensions to multivariate analysis of variance. *Journal of the Royal Statistical Society: Series C*, **48**, 505–519.

Gower, J. C. and Legendre, P. (1986). Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, **3**, 5–48.

Green, B. (1952). The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika*, **17**, 429–440.

Griffiths, C. and Barker, J. (2007). Pathogenesis and clinical features of psoriasis. *Lancet*, **370**, 263–271.

Guvenir, H., Demiroz, G. and Ilter, N. (1998). Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. *Artificial Intelligence in Medicine*, **13**, 147–165.

Hartigan, J. A. (1985). Statistical theory in clustering. *Journal of Classification*, **2**, 63–76.

Hartigan, J. A. and Wong, M. A. (1979). A k-means clustering algorithm. *Applied Statistics*, **28**, 126–130.

Heard, N. A., Holmes, C. C., Stephens, D. A., Hand, D. J. and Dimopoulos, G (2005). Bayesian coclustering of *Anopheles* gene expression time series: study of immune defense response to multiple experimental challenges. *Proceedings of the National Academy of Sciences of the United States of America*, **102**, 16939–16944.

Herwig, R., Poustka, A., Meuller, C., Lehrach, H. and O'Brien, J. (1999). Large-scale clustering of cDNA–fingerprinting data. *Genome Research*, **9**, 1093–1105.

Hettich, S. and Bay, S. D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Department of Information and Computer Science. University of California at Irvine, Irvine, USA.

Heyer, L., Kruglyak, S. and Yooseph, S. (1999). Exploring expression data: identification, and analysis of coexpressed genes. *Genome Reseach*, **9**, 1106–1115.

Humbert, J. (1988). *Mitología griega y romana*. Editorial Gustavo Gili, Madrid.

Hurley, J. and Catell, R. (1962). The procrustes program: producing direct rotation to test a hypothesized factor structure. *Behavioral Science*, **7**, 258–262.

Irigoien, I. and Arenas, C. (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units. *Statistics in Medicine*, **27**, 2948–2973.

Irigoien, I., Fernández, E., Vives, S. and Arenas, C. (2006). Clapper: A clustering algorithm based on path-distances for experimental replicates. *Advances and Applications in Statistics*, **6**, 379–400.

Irigoien, I., Fernández, E., Vives, S. and Arenas, C. (2008). Clum: A cluster

program for analyzing microarray data. *Russian Journal of Genetics*, **44**, 993–996.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Englewood Cliffs, USA.

Kaufman, L. and Rousseeuw, P. (1990). *Finding groups in data. An introduction to cluster analysis*. Wiley and Sons, New York.

Krimbas, C. B. (1993). *D. subobscura: Biology, Genetics and Inversion polymorphism*. Verlag Dr. Kovac, Hamburg.

Kristof, W. and Wingersky, B. (1971). Generalization of the orthogonal procrustes rotation procedure to more than two matrices. *Proceedings of the 79th Annual Convention of the American Psychologycal Association*, **6**, 89–90.

Krzanowski, W. J. (2004). Biplots for multifactorial analysis of distance. *Biometrics*, **60**, 517–524.

Krzanowski, W. J. and Lai, Y. (1988). A criterion for determining the number of groups in a dataset using sum of squares clustering. *Biometrics*, **44**, 23–34.

Krzanowski, W. J. and Marriott, F. H. C. (1994). *Multivariate analysis. Part 1: Distributions, Ordination and Inference*. Kendall's Library of Statistics. Edward Arnold, London.

Kulesh, D. A., Clive, D. R., Zarlenga, D. S. and Greene, J. J. (1987). Identification of Interferon-Modulated Proliferation-Related cDNA Sequences. *Proceedings of the National Academy of Sciences of the United States of America*, **84**, 8453–8457.

Lance, G. N. and Williams, W. T. (1967). A general theory of classification sorting strategies: 1. Hierarchical systems. *Computation Journal*, **9**, 373–380.

Legendre, P. and Anderson, M. J. (1999). Distance-based redundancy analysis: Testing multispecies responses in multifactorial ecological experiments. *Ecological Monographs*, **48**, 505–519.

Lingoes, J. C. (1971). Some boundary conditions for a monotone analysis of symmetric matrices. *Psychometrika*, **36**, 195–203.

Lockhart, D., Dong, H., Byrne, M., Follettie, M., Gallo, M. V., Chee, M. S., Mittmann, M., Wang, C., Kobayashi, M., Horton, H. and Brown, E. L. (1996). Expression monitoring by hybridization to high density oligonucleotide arrays. *Nature Biotechnology*, **14**, 1675–1680.

Lund, J., Tedesco, P., Duke, K., Wang, J., Kim, S. K. and Hohnson, T. E. (2002). Transcriptional profile of aging in *C. elegans*. *Current Biology*, **12**, 1566–1573.

Luo, F., Khan, L., Bastani, F., Yen, I. L. and Zhou, J. (2004). A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles. *Bioinformatics*, **20**, 2605–2617.

Luukka, P. (2007). Similarity classifier using similarity measure derived from Yu's norms in classification of medical data sets. *Computers in Biology and Medicine*, **37**, 1133–1140.

Ma, P., Castillo-Davis, C. I., Zhong, W. and Liu, J. S. (2006). A data-driven clustering method for time course gene expression data. *Nucleic Acids Research*, **34**, 1261–1269.

Maxam, A. M. and Gilbert, W. (1977). A new method for sequencing DNA. *Proceedings of the National Academy of Sciences of the United States of America*, **74**, 560–564.

McDonald, L. L., Lowe, V. W., Smidt, R. K. and Meister, K. A. (1976). A preliminary test for discriminant analysis based on small samples. *Biometrics*, **32**, 417–422.

McLachlan, G. J. (1982). On the bias and variance of some proportion estimators. *Communications in Statistics, Simulation and Computation*, **11**, 715–736.

McLachlan, G. J. and Basford, K. (1988). *Mixture models: inference and applications to clustering*. Marcel Dekker, New York.

Mestres, F., Serra, L. and Ayala, F. J. (1995). Colonization of the Americas by *D. subobscura*: Lethal-gene allelism and association with chromosomal arrangements. *Genetics*, **140**, 1297–1305.

Mestres, F., Balanyà, J., Pascual, M., Arenas, C., Solé, E. and Serra, L. (2005). Lethal genes and the colonization of America by *D. subobscura*. *Current Topics in Genetics*, **1**, 31–57.

Mestres, F., Balanyà, J., Pascual, M., Arenas, C., Gilchrist, G. W., Huey, R. B. and Serra, L. (2008). Evolution of Chilean colonizing populations of *D. subobscura*: Lethal-genes and chromosomal arrangements. *Genetica*, DOI 10.1007/s10709-008-9298-y.

Milligan, G. W. and Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, **50**, 159–179.

Möller-Levet, C., Cho, K. H. and Wolkenhauer, O. (2003). DNA microarray data clustering based on temporal variation: FCV with TSD preclustering. *Applied Bioinformatics*, **2**, 35–45.

Montanari, A. and Mignari, S. (1994). Notes on the bias of dissimilarity indices for incomplete data sets: the case of archaelogical classifications. *Qüestiió*, **18**, 39–49.

Moody, S. (2007). *Principles of Developmental Genetics*. Academic Press, Burlington, USA.

Mosier, C. (1939). Determining a simple structure when loadings for certain tests are known. *Psychometrika*, **4**, 149–162.

Nüslein-Volhard, C. (2006). *Coming to life: How genes drive development*. Yale University Press, USA.

Peddada, S., Lobenhofer, E., Li, L., Afshari, C., Weinberg, C. and Umbach, D. (2003). Gene selection and clustering for time-course and dose-response microarray experiments using order-restricted inference. *Bioinformatics*, **19**, 834–841.

Plump, A. S., Erickson, S. K., Weng, W., Partin, J. S., Breslow, J. L. and Williams, D. L. (1996). Apolipoprotein A-I is required for cholesteryl ester accumulation in steroidogenic cells and for normal adrenal steroid production. *The Journal of Clinical Investigation*, **97**, 2660–2671.

Prevosti, A., Serra, L., Aguadé, M., Ribó, G., Mestres, F., Balanyà, J. and Monclús, M. (1985). Colonization and establishment of the Palearctic species *D.*

subobscura in North and South America. In: *Evolutionary Biology of Transient Unstable Populations*, pp. 114–129. Fontdevila, A. (ed.). Springer-Verlag, Berlin.

Pui, C. and Evans, W. (1998). Drug therapy: Acute lymphoblastic leukemia. *The New England Journal of Medicine*, **339**, 605–615.

Ramoni, M. F., Sebastiani, P. and Kohane, S. K. (2002). Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences of the United States of America*, **99**, 9121–9126.

Rao, C. R. (1962). Use of discriminant and allied functions in multivariate analysis. *Sankhyā. The Indian Journal of Statistics: Series A*, **24**, 149–154.

Rao, C. R. (1982). Diversity: its measurement, decomposition, apportionment and analysis. *Sankhyā. The Indian Journal of Statistics: Series A*, **44**, 1–22.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, **20**, 53–65.

Sanger, F. and Thompson, E. O. P. (1963). The amino acid sequence in the glycyl chain of insulin. *Biochemical Journal*, **53**, 353–374.

Sanger, F. and Tuppy, H. (1961). The amino acid sequence in the phenylalanyl chain of insulin. *Biochemical Journal*, **49**, 463–490.

Sanger, F., Nicklen, S. and Coulson, A. R. (1977). DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, **74**, 5463–5467.

Schlossman, S. F., Chess, L., Humphreys, R. and Strominger, J. (1976). Distribution of Ia-like molecules on the surface of normal and leukemic human cells. *Proceedings of the National Academy of Sciences of the United States of America*, **73**, 1288–1292.

Schönemann, P. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, **31**, 1–10.

Schönemann, P. (1968). On two sided orthogonal procrustes problems. *Psychometrika*, **33**, 19–33.

Schönemann, P. and Carroll, R. (1970). Fitting one matrix to another under choice of a central dilation and a rigid motion. *Psychometrika*, **35**, 245–255.

Sibson, R. (1978). Studies in the robustness of multidimensional scaling: Procrustes statistics. *Journal of the Royal Statistical Society: Series B*, **40**, 234–238.

Slonim, D. K. (2002). From patterns to pathways: gene expression data analisis comes of age. *Nature Genetics supplement*, **32**, 502–508.

Solé, E., Mestres, F., Balanyà, J., Arenas, C. and Serra, L. (2000). Colonization of America by *D. subobscura*: Spatial and temporal lethal-gene allelism. *Hereditas*, **133**, 65–72.

Storey, J. D. and Tibshirani, R. (2003). Statistical significance for genome-wide studies. *Proceedings of the National Academy of Sciences of the United States of America*, **100**, 9440–9445.

Storey, J. D., Xiao, W., Leek, J. T., Tompkins, R. G. and Davis, R. W. (2005). Significance analysis of time course microarray experiments. *Proceedings of the National Academy of Sciences of the United States of America*, **102**, 12837–12842.

Tan, M., Broach, J. and Floudas, C. (2007). A novel clustering approach and prediction of optimal number of clusters: global optimum search with enhanced positioning. *Journal of Global Optimization*, **39**, 323–346.

Tavazoie, S., Hughes, J. D., Campbell, M. J., Cho, R. J. and Church, G. M. (1999). Systematic determination of genetic network architecture. *Nature Genetics*, **22**, 281–285.

ten Berge, J. (1977). Orthogonal procrustes rotation for two or more matrices. *Psychometrika*, **42**, 267–276.

ten Berge, J. (2006). The rigid orthogonal procrustes rotation problem. *Psychometrika*, **71**, 201–205.

Tibshirani, R., Walther, G. and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B*, **63**, 411–423.

Tomancak, P., Beaton, A., Weiszmann, R., Kwan, E., Shu, S., Lewis, S. E., Richards, S., Ashburner, M., Hartenstein, V., Celniker, S. E. and Rubin, G. M. (2002). Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology*, **3**, 1–14.

Tseng, G. C. (2005). Tight clustering: A resampling-based approach for identifying stable and tight patterns in data. *Biometrics*, **61**, 10–16.

Varma, S. and Simon, R. (2004). Iterative class discovery and feature selection using Minimal Spanning Trees. *BMC Bioinformatics*, **5**, 126–134.

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J. et al., (2001). The sequence of the human genome. *Science*, **291**, 1304–1351.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, **58**, 236–244.

Wen, X., Fuhrman, S., Michaels, G. S., Carr, D. B., Smith, S. and Somogayi, R. (1998). Large-scale temporal gene expression mapping of central nervous system development. *Proceedings of the National Academy of Sciences of the United States of America*, **95**, 334–339.

Wieschaus, E. and Nüsslein-Volhard, C. (1998). Looking at embryos. In: *Drosophila. Practical Approach*, pp. 179–214. Roberts, D. B. (ed.), IRL Press, Oxford.

Williamson, R., Lee, D., Hagaman, J. and Maeda, N. (1992). Marked reduction of high density lipoprotein cholesterol in mice genetically modified to lack apolipoprotein A-I. *Proceedings of the National Academy of Sciences of the United States of America*, **89**, 7134–7138.

Xu, Y., Olman, V. and Xu, D. (2002). Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, **18**, 536–545.

Yan, M. and Ye, K. (2007). Determining the number of clusters using the weighted gap statistic. *Biometrics*, **63**, 1031–1037.

Yang, Y. H., Dudoit, S., Luu, P., Lin, D. M., Peng, V., Ngai, J. and Speed, T. P. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple lide systematic variation. *Nucleic Acids Research*, **30**, e15.

Zhao, L., Prentice, R. and Breeden, L. (2001). Statistical modeling of large microarray data sets to identify stimulus–response profiles. *Proceedings of the National Academy of Sciences of the United States of America*, **98**, 5631–5636.

Zivanovic, G., Andjelkovic, M. and Marinkovic, D. (2000). Genetic load and coadpatation of chromosomal inversion. II. O-chromosomes in *D. subobscura* populations. *Hereditas*, **133**, 105–113.

Anexo A

Manual del paquete «DBmethods» en R

A continuación se adjunta el manual del paquete «DBmethods» que se ofrece dentro del software de libre distribución R con la descripción detallada de cada función principal creada para la posible aplicación de los métodos descritos en esta memoria.

`CLAPPER` *CLAPPER: Clustering Based on the Path Distance*

Description

Given the distance matrix of n objects, `CLAPPER` calculates a partition of the objects in two clusters C1 and C2. Replication of measurements for each object can be included.

Usage

```
CLAPPER(d, renom, replicates = FALSE, DV = NULL, coef = 2/3)
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.
`renom`	names for the objects to be clustered.
`replicates`	logical indicating whether the replication of the data is or not considered. The default value is FALSE.
`DV`	only for `replicates = TRUE`. Vector of length n containing the measurements of variability between replicates.
`coef`	To compute the `alpha` value of the algorithm, such as, `alpha = coef * (max(DV) - min(DV))`. The default value is 2/3.

Value

It is a list with components:

`n1`	number of objects in cluster C1.
`g1`	objects assigned to cluster C1.
`dp1`	path distance of cluster C1.
`n2`	number of objects in cluster C2.
`g2`	objects assigned to cluster C2.
`dp2`	path distance of cluster C2.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Irigoien, I., Fernández, E., Vives, S. and Arenas, C. (2006). CLAPPER: a clustering algorithm based on path-distances for experimental replicates. *Advances and Applications in Statistics*, **6**, 377–398.

Irigoien, I., Fernández, E., Vives, S. and Arenas, C. (2008). CLUM: A cluster program for analyzing microarray data. *Russian Journal of Genetics*, **44**, 993–996.

See Also

path.d

Examples

```
# generate 2 clusters, each of them with 20 objects in dimension 5.
mu1 <- sample(1:10, 5, replace=TRUE)
x1  <- matrix(rnorm(20*5, mean=mu1, sd=1), ncol=5, byrow=TRUE)
mu2 <- sample(1:10, 5, replace=TRUE)
x2  <- matrix(rnorm(20*5, mean=mu2, sd=1), ncol=5, byrow=TRUE)

x <- rbind(x1,x2)

d <- dist(x)
n <- 1:40 # names of the objects

CLAPPER(d, n )

# Simulated data representing 3 replicates of the expression measurement for 11 genes
x <- matrix( c(3.88,  3.78,  5.53,
 3.01,  7.34,  1.40,
17.36, 16.64, 18.86,
14.10, 20.28, 10.31,
 8.32, 10.35,  1.56,
33.65, 37.11, 44.80,
37.66, 42.06, 43.47,
44.98, 44.83, 45.64,
44.97, 42.13, 42.86,
45.56, 45.45, 50.31,
60.72, 60.25, 59.93), nrow=11, byrow=TRUE)

colnames(x) <- c("Replication1", "Replication2", "Replication3")
rownames(x) <- 1:11

d <- dist(x) # Distance between pairs of objects

v <- apply(x, 1, sd) # Variability between replicates

CLAPPER(d,  rownames(x), replicates=TRUE, DV= v)
```

`GEVAdivisive` *Hierachical Divisive Clustering*

Description

Given a dissimilarity matrix between n objects `GEVAdivisive` divides the objects in two clusters C1 and C2 such that the sum of the weighted geometric variabilities of the clusters is minimum.

Usage

```
GEVAdivisive(d, renom = NULL)
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.
`renom`	vector of length n with names for the objects to be clustered.

Value

It is a list with components:

`VT`	sum of the weighted geometric variabilities of the clusters.
`n1`	number of objects in cluster C1.
`g1`	objects assigned to cluster C1.
`n2`	number of objects in cluster C2.
`g2`	objects assigned to cluster C2.

Note

Warning: It is a divisive technique and it could consume a lot of time according to the data size.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

Cuadras, C.M. (1992). Some examples of distance based discrimination. *Biometrical Letters*, **29**, 3–20.

See Also

GEVAjoining, vgeo

Examples

```
# Prevosti's distances between Drosophila species according to their daily activity
n <- 8
species <- c("immigrans", "melanogaster", "phalerata", "simulans", "subobscura",
           "testecea", "hydei", "cameraria")
distances <- c(0.35, 0.48, 0.25, 0.46, 0.42, 0.46, 0.68, 0.52, 0.31, 0.40, 0.47,
             0.48, 0.65, 0.53, 0.49, 0.40, 0.65, 0.71, 0.37, 0.46, 0.51, 0.67,
             0.52, 0.63, 0.58, 0.54, 0.63, 0.63)
d <- matrix(0, n,n)
aux <- 0
for (i in 1:(n-1)){
    for (j in (i+1):n){
        aux <- aux +1
        d[i,j] <- distances[aux]
        d[j,i] <- d[i,j]
    }
}
rownames(d) <- species
colnames (d) <- rownames(d)

# First divisive partition
dp <- GEVAdivisive(d, species)
```

`GEVAjoining` *Agglomerative Hierarchical Clustering GEVAjoining*

Description

`GEVAjoining` performs a hierarchical agglomerative clustering merging each time the two clusters, such that the geometric variability of their union is the minimum among all the possible unions in a given partition.

Usage

```
GEVAjoining(d)
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.

Value

An object of class `hclust` which describes the tree produced by the clustering process. The object is a list with components:

`merge`	an n-1 by 2 matrix. Row i of merge describes the merging of clusters at step i of the clustering. If an element j in the row is negative, then observation -j is merged at this stage. If j is positive then the merge is with the cluster formed at the (earlier) stage j of the algorithm. Thus negative entries in merge indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.
`height`	a set of n-1 non-decreasing real values. The clustering height: the value of the criterion associated with the clustering method for the particular agglomeration.
`order`	a vector giving the permutation of the original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix merge will not have crossings of the branches.

Author(s)

Based on the `hierclust.R` function by F. Murtagh.

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsiatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d' Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

Cuadras, C.M. (1992). Some examples of distance based discrimination. *Biometrical Letters*, **29**, 3–20.

See Also

GEVAdivisive, vgeo

Examples

```
# Prevosti's distances between Drosophila species according to their daily activity
n <- 8
species <- c("immigrans", "melanogaster", "phalerata", "simulans", "subobscura",
           "testecea", "hydei", "cameraria")
distances <- c(0.35, 0.48, 0.25, 0.46, 0.42, 0.46, 0.68, 0.52, 0.31, 0.40, 0.47,
             0.48, 0.65, 0.53, 0.49, 0.40, 0.65, 0.71, 0.37, 0.46, 0.51, 0.67,
             0.52, 0.63, 0.58, 0.54, 0.63, 0.63)
d <- matrix(0, n,n)
aux <- 0
for (i in 1:(n-1)){
    for (j in (i+1):n){
        aux <- aux +1
        d[i,j] <- distances[aux]
        d[j,i] <- d[i,j]
    }
}
rownames(d) <- species
colnames (d) <- rownames(d)

# Agglomerative clustering
hc <- GEVAjoining(d)
plot(hc, hang=-0.1, labels=species)
```

`INCAnumclu` *Estimation of Number of Clusters in Data*

Description

`INCAnumclu` helps to estimate the number of clusters in a dataset.

Usage

```
INCAnumclu(d, pert_clus)
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.
`pert_clus`	vector of length n indicating which group each individual belongs to. The default value indicates that there is only one group in data.

Value

It is a list with components:

`well_class`	vector indicating the number of objets well classified in its cluster.
`Ni_cluster`	vector indicating the cluster size.
`Total`	percentage of objects well classified in the partition defined by `pert_clus`.

Note

In order to get a correct geometrical interpretation of INCA statistic, it is convenient to verify whether the distance matrix `d` is Euclidean.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Irigoien, I. and Arenas, C. (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units. *Statistics in Medicine*, **27**, 2948–2973.

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

See Also

`estW`, `INCAtest`

Examples

```
#generate 3 clusters of, each of them with 20 objects in dimension 5.
mu1 <- sample(1:10, 5, replace=TRUE)
x1 <- matrix(rnorm(20*5, mean = mu1, sd = 1),ncol=5, byrow=TRUE)
mu2 <- sample(1:10, 5, replace=TRUE)
x2 <- matrix(rnorm(20*5, mean = mu2, sd = 1),ncol=5, byrow=TRUE)
mu3 <- sample(1:10, 5, replace=TRUE)
x3 <- matrix(rnorm(20*5, mean = mu3, sd = 1),ncol=5, byrow=TRUE)
x <- rbind(x1,x2,x3)

# calculate Euclidean distance between them
d <- dist(x)

# given the right partition, calculate the percentage of well classified objects
partition <- c(rep(1,20), rep(2,20), rep(3,20))
INCAnumclu(d, partition)

# In order to estimate the number of cluster in data, try several
#  partitions and compare the results
library(cluster)
T <- rep(NA, 5)
for (l in 2:5){
        part <- pam(d,l)$clustering
        T[l] <- INCAnumclu(d,part)$Total
}

plot(T, type="b",xlab="Number of clusters", ylab="INCA", xlim=c(1.5, 5.5))
```

`INCAtest` *INCA Test*

Description

Given the distances from one specific individual to the n individuals organized in several groups, `INCAtest` performs a typicality test, the so-called INCAtest.

Usage

```
INCAtest(d, pert, d_test, np = 1000, alpha = 0.05, P = 1)
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.
`pert`	vector of length n indicating which group each individual belongs to.
`d_test`	vector of length n containing the distances from the specific individual to the rest of individuals.
`np`	sample size for the bootstrap sample.
`alpha`	fixed level for the test.
`P`	the bootstrap procedure is repeated 10*P times.

Value

`StatisticW0`	value of the INCA statistic.
`ProjectionsU`	values of statistics measuring the projection from the specific object to each considered group.
`Percentage_under_alpha`	percentage of times the INCA test has been rejected for the specified value of `alpha`.
`alpha`	specified value of the level of the test.

Note

Take into account that calculation of the null distribution of the INCA statistic could consume a lot of time according to the data size.

In order to assure that the INCA statistic is meaningful, check whether the distance matrix is Euclidean.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Irigoien, I. and Arenas, C. (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units. *Statistics in Medicine*, **27**, 2948–2973.

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

See Also

estW, INCAnumclu

Examples

```
#generate 3 clusters of, each of them with 20 objects in dimension 5.
mu1 <- sample(1:10, 5, replace=TRUE)
x1 <- matrix(rnorm(20*5, mean = mu1, sd = 1),ncol=5, byrow=TRUE)
mu2 <- sample(1:10, 5, replace=TRUE)
x2 <- matrix(rnorm(20*5, mean = mu2, sd = 1),ncol=5, byrow=TRUE)
mu3 <- sample(1:10, 5, replace=TRUE)
x3 <- matrix(rnorm(20*5, mean = mu3, sd = 1),ncol=5, byrow=TRUE)
x <- rbind(x1,x2,x3)

# calculate euclidean distance between them
d <- dist(x)
# given the right partition
partition <- c(rep(1,20), rep(2,20), rep(3,20))

# characteristics of a specific object from group 1
x0 <-  matrix(rnorm(1*5, mean = mu1, sd = 1),ncol=5, byrow=TRUE)

# distances between  object x0 and the rest of objects in data x
dx0 <- rep(0,60)
for (i in 1:60){
        dif <-x0-x[i,]
        dx0[i] <- sqrt(sum(dif*dif))
}

INCAtest(d, partition, dx0, np=10)

# characteristics of a specific object, likely, from a new group
x0 <-  matrix(rnorm(1*5, mean = sample(1:10, 5, replace=TRUE),
        sd = 1), ncol=5, byrow=TRUE)

# distances between  object x0 and the rest of objects in data x
dx0 <- rep(0,60)
for (i in 1:60){
        dif <-x0-x[i,]
        dx0[i] <- sqrt(sum(dif*dif))
}

INCAtest(d, partition, dx0, np=10)
```

`dbhatta` *Bhattacharyya Distance*

Description

`dbhatta` calculates the Bhattacharyya distance matrix between pair of objects. This distance is adequate when each object is characterized by an m-dimensional vector $p_1, ..., p_m$ in the space whose coordinates are the generated frequencies with $p_1, ..., p_m >= 0$ and $p_1 + ... + p_m = 1$.

Usage

```
dbhatta(x)
```

Arguments

`x`	matrix with frequencies for each object in its rows.

Value

It is a matrix with Bhattacharyya distances between pairs of `x` rows.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Bhattacharyya, A. (1946). On a measure of divergence of two multinomial populations. *Sankhya: The Indian Journal of Statistics, Series A*. **14**, 177-136.

See Also

`dist`, `dmahal`, `dgower`

Examples

```
#Generate 10 objects in dimension 4, as observed frequencies out of M=30 trials
n <- 10
f <- matrix(0, n, 4)
M <- 30
for (i in 1:n){
    f[i,] <- tabulate(sample(1:4, M, replace=TRUE))/M
}
```

```
# Bhattacharyya distances between pairs
d <- dbhatta(f)
```

`dcor` *Correlation Distance*

Description

`dcor` calculates the correlation distance matrix between pair of objects as $d = \sqrt{1-r}$.

Usage

```
dcor(x)
```

Arguments

`x`	data matrix with objects in its rows.

Value

It is a matrix with Correlations distances between pairs of `x` rows.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Gower, J.C. (1985). Measures of similarity, dissimilarity and distance. In: *Encyclopedia of Statistical Sciences*, volume **5**, 397–405. J. Wiley and Sons.

See Also

`dist`, `dmahal`, `dgower`, `dbhatta`

Examples

```
#Generate 10 objects in dimension 8
n <- 10
mu <- sample(1:10, 8, replace=TRUE)
x <- matrix(rnorm(n*8, mean=mu, sd=1), nrow=n, byrow=TRUE)

# Correlation distances between pairs
d <- dcor(x)
```

`deltas` *Distance Between Groups*

Description

`deltas` calculates the distance between each pair of groups in data, based on distances.

Usage

```
deltas(d, pert = "onegroup")
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.
`pert`	vector of length n indicating which group each individual belongs to. The default value indicates that there is only one group in data.

Value

It is a matrix containing the distances between each pair of groups.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

Cuadras, C.M., Fortiana, J. and Oliva, F. (1997). The proximity of an individual to a population with applications in discriminant analysis. *Journal of Classification*, **14**, 117–136.

See Also

`vgeo`, `proxi`

Examples

```
data(iris)
d <- dist(iris[,1:4])
deltas(d,iris[,5])
```

`dgower` *Gower Distance for Mixed Variables*

Description

`dgower` calculates de Gower distance for mixed variables.

Usage

```
dgower(x, type = list())
```

Arguments

`x`	data matrix.
`type`	it is a list with components `cuant`, `bin`, `nom`. Each component must indicate the column position of the quantitative, binary or nominal variables, respectively.

Details

The distance between two pairs of objects i and j is obtained as $\sqrt{(2(1 - s_{ij}))}$ where s_{ij} is the Gower's similarity coefficient for mixed data. This function allows to include missing values (as `NA`) and therefore calculates distances based on the Gower's weighted similarity coefficient.

Value

It is a distance matrix.

Note

There is `daisy` in library `cluster` which calculates another version of the Gower distance. `daisy` calculates the distance between two pairs of objects i and j as $1 - s_{ij}$, where s_{ij} is the Gower's similarity coefficient.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Gower, J.C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, **27**, 857–871.

See Also

dist, dmahal, dbhatta

Examples

```
#Generate 10 objects in dimension 6
# Quantitative variables
mu <- sample(1:10, 2, replace=TRUE)
xc <- matrix(rnorm(10*2, mean = mu, sd = 1), ncol=2, byrow=TRUE)

# Binary variables
xb <- cbind(rbinom(10, 1, 0.1), rbinom(10, 1, 0.5), rbinom(10, 1, 0.9))

# Nominal variables
xn <- matrix(sample(1:3, 10, replace=TRUE), ncol=1)

x <- cbind(xc, xb, xn)

# Distances
d <- dgower(x, type=list(cuant=1:2, bin=3:5, nom=6))
```

`dmahal` *Mahalanobis Distance*

Description

`dmahal` calculates the Mahalanobis distance between pairs of objects.

Usage

```
dmahal(datos, S)
```

Arguments

`datos`	data matrix.
`S`	covariance matrix.

Value

It is a matrix with Mahalanobis distances between pairs of `x` rows.

Note

The function `dmahal` calculates the Mahalanobis function between pairs of rows of the data matrix, whereas the function `mahalanobis` calculates the Mahalanobis distance between each row of the data matrix and the center of data.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Peña D. (2002) *Analisis de Datos Multivariantes*. Mc Graw Hill.

Everitt B. S. and Dunn G. (2001) *Applied Multivariate Data Analysis*. 2 edition, Edward Arnold, London.

See Also

`dist`, `dbhatta`, `dgower`

Examples

```
#Generate 10 objects in dimension 2
mu <- rep(0, 2)
Sigma <- matrix(c(10,3,3,2),2,2)

x <- mvrnorm(n=10, rep(0, 2), Sigma)

d <- dmahal(x, Sigma)
```

`estW` *INCA Statistic*

Description

Given the distances from one specific individual x_0 to the n individuals organized in $C_1, ..., C_k$ groups, `estW` calculates the INCA statistic $W(x_0)$ as well as the relied statistics $U_i = \phi(x_0, C_i) - W(x_0)$, where $\phi(x_0, C_i)$ is the proximity function from individual x_0 to group C_i (i=1, ..., k).

Usage

```
estW(d, dx0, pert = "onegroup")
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.
`dx0`	vector of length n containing the distances from the specific individual to the rest of individuals.
`pert`	vector of length n indicating which group each individual belongs to. The default value indicates that there is only one group in data.

Value

It is a list with components:

`Wvalue`	is the INCA statistic $W(x_0)$.
`Uvalue`	is the vector containing the statistics U_i, i=1, ..., k.

Note

In order to get a correct geometrical interpretation of $W(x_0)$, it is convenient to verify whether the distance matrix `d` is Euclidean.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Irigoien, I. and Arenas, C. (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units. *Statistics in Medicine*, **27**, 2948–2973.

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

See Also

vgeo, proxi , deltas

Examples

```
data(iris)
d <- dist(iris[,1:4])

# characteristics of a specific flower (likely group 1)
x0 <- c(5.3, 3.6, 1.1, 0.1)
# distances between  flower x0 and the rest of flowers in iris
dx0 <- rep(0,150)
for (i in 1:150){
        dif <-x0-iris[i,1:4]
        dx0[i] <- sqrt(sum(dif*dif))
}
estW(d, dx0, iris[,5])
```

path.d *Path Distance*

Description

Given a distance matrix between each pair of objects in a group, `path.d` calculates the path distance of the group. It is a measure of the variability of the cluster.

Usage

```
path.d(d)
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.

Value

Path distance of the group (numeric).

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Irigoien, I., Fernández, E., Vives, S. and Arenas, C. (2008). CLUM: A cluster program for analyzing microarray data. *Russian Journal of Genetics*, **44**, 993–996.

Irigoien, I., Fernández, E., Vives, S. and Arenas, C. (2006). CLAPPER: a clustering algorithm based on path-distances for experimental replicates. *Advances and Applications in Statistics*, **6**, 377–398.

See Also

CLAPPER

Examples

```
library(datasets)
d <- eurodist

# Given a distance matrix between objects,
path.d(d) #so, fixed a city, we can reach any other city in steps of maximum length 817.
```

proc2 *Modified Procrustes Statistic*

Description

In the context of microarray time course experiments, given two time course measurements, `proc2` evaluates the similarity between the two corresponding configurations, by means of the usual operations in Procrustes Analysis, such as translation and scaling, the rotation being excluded.

Usage

```
proc2(X, Y)
```

Arguments

`X`	vector containing all the observed measurements along the time course experiment for the first configuration.
`Y`	vector containing all the observed measurements along the time course experiment for the second configuration.

Details

Each configuration is arranged in a two columns matrix where in the first column are the time points 1, ..., T and in the second column the observed measurements X=(x1, ..., xT).

Value

The procrustes statistic value is returned.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Gower, J. C. and Dijksterhuis, G. B. (2004). *Procrustes Problems.* Oxford University Press.

See Also

`prock`, `serieprock`

Examples

```
# Given  two hypothetic time course profiles with 12 time points
x <- c(13.1,  12.9,  10.9,  8.9,  8.8,  8.8,  8.9,  8.8,  8.9,  9.1,  9.3,  9.4)
y <- c(12.8,  13.0 , 11.0,  8.9,  9.0,  8.9,  8.9,  8.9,  8.9,  9.4,  9.4,  9.3)

# Graphical representation
matplot(matrix(data=c(x,y), nrow=12, byrow=FALSE), type="b")

# Similarity between them (near 0, strong similarity)
proc2(x,y)
```

`prock` *Modified Generalized Procrustes Analysis*

Description

In the context of microarray time course experiments, given a number of time course measurements higher than two, `prock` evaluates the similarity between the corresponding configurations by means of the usual operations in Procrustes Analysis, such as translation and scaling, the rotation being excluded.

Usage

```
prock(dat)
```

Arguments

`dat`	is a matrix containing in each row all the observed measurements along the time course experiment corresponding to each configuration.

Details

Each configuration is arranged in a two columns matrix where in the first column are the time points 1, ..., T and in the second column the observed measurements X=(x1, ..., xT).

Value

It is a list with components:

`Mmean`	the weighted procrustes mean configuration.
`scales`	scaling coefficients.
`Mvalue`	the generalized procrustes statistic.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Gower, J. C. and Dijksterhuis, G. B. (2004). *Procrustes Problems.* Oxford University Press.

See Also

`proc2`, `serieprock`

Examples

```
# Three hypothetic time course profiles with 12 time points
x <- c(13.1,  12.9,  10.9,  8.9,  8.8,  8.8,  8.9,  8.8,  8.9,  9.1,  9.3,  9.4,
12.8,  13.0 , 11.0,  8.9,  9.0,  8.9,  8.9,  8.9,  8.9,  9.4,  9.4,  9.3,
13.0,  12.9,  11.4,  8.9,  8.7,  8.8,  8.8,  8.9,  9.2,  9.2,  9.3,  9.3)
x <- matrix(x, nrow=3, ncol=12, byrow=TRUE)

# Graphical representation
matplot(t(x), type="b")

# Similarity between them (near 0, strong similarity)
prock(x)
```

proxi *Proximity Function*

Description

Given the distances from one specific individual to the n individuals organized in several groups, `proxi` calculates the proximity function from the specific individual to the groups in data.

Usage

```
proxi(d, dx0, pert = "onegroup")
```

Arguments

`d`	distance matrix or a `dist` object for the n individuals.
`dx0`	vector of length n containing the distances from the specific individual to the rest of individuals.
`pert`	vector of length n indicating which group each individual belongs to. The default value indicates that there is only one group in data.

Value

It is a vector containing the proximity function from the specific individual to each group.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

Cuadras, C.M., Fortiana, J. and Oliva, F. (1997). The proximity of an individual to a population with applications in discriminant analysis. *Journal of Classification*, **14**, 117–136.

See Also

`vgeo`, `deltas`

Examples

```
data(iris)
d <- dist(iris[,1:4])

# characteristics of a specific flower (likely group 1)
x0 <- c(5.3, 3.6, 1.1, 0.1)
# distances between  flower x0 and the rest of flowers in iris
dx0 <- rep(0,150)
for (i in 1:150){
        dif <-x0-iris[i,1:4]
        dx0[i] <- sqrt(sum(dif*dif))
}

proxi(d, dx0, iris[,5])

# characteristics of a specific flower (likely group 2)
x0 <- c(6.4, 3.0, 4.8, 1.3)
# distances between  flower x0 and the rest of flowers in iris
dx0 <- rep(0,150)
for (i in 1:150){
        dif <-x0-iris[i,1:4]
        dx0[i] <- sqrt(sum(dif*dif))
}

proxi(d, dx0, iris[,5])
```

`serieprock` *Modified Generalized Procrustes Statistic*

Description

In the context of microarray time course experiments with replicates, `serieprock` calculates for each gene the similarity between its replicates and sorts them decreasingly. The similarity is calculated by means of the usual operations in Procrustes Analysis, such as translation and scaling, the rotation being excluded.

Usage

```
serieprock(dat, nrep, g, null.p = FALSE)
```

Arguments

`dat`	is a matrix containing in each row all the observed measurements along the time course experiment corresponding to each configuration. All replication measurements corresponding to the same gene must be in consecutive rows. All genes must have the same number of replicated configurations. Given n genes with R replicates and in a time course experiment with T time points, `dat` must be a (n*R) * T dimension matrix.
`nrep`	number of replications.
`g`	identification name for each gene.
`null.p`	logical value indicating whether the null profile should be included or not. The default value is FALSE.

Details

Each configuration is arranged in a two columns matrix where in the first column are the time points 1, ..., T and in the second the observed measurements X=(x1, ..., xT). The null profile is defined as a flat profile by the configuration matrix (1, ..., T; 1, ..., 1).

Value

`Mvalue`	vector containing the modified generalized procrustes statistics values in decreasing order.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Gower, J. C. and Dijksterhuis, G. B. (2004) *Procrustes Problems.* Oxford University Press.

See Also

proc2, prock, varepli

Examples

```
# Generate 10 hypothetic time course profiles with T=12 time points, with 3 replicates.
T <- 12
R <- 3
x <- matrix(0, nrow=10*R, ncol=T)
for (i in 1: 10){
    m <- sample(1:10, T, replace=TRUE)
    x[(R*(i-1))+ 1,] <- m
    for (r in 2:R){
        for (r in 2:R){
           x[(R*(i-1))+ r,] <- m + runif(T, -0.5, 0.5)
        }
    }
}
id <- LETTERS[1:10] # identification of genes

serieprock(dat=x, nrep=3,g=id)

# Including the null profile
serieprock(dat=x, nrep=3,g=id, null.p=TRUE)
```

`varepli` *Variability Between Replicates*

Description

In the context of microarray time course experiments with replicates, `varepli` evaluates the variability between the replicates for each gene using bootstrap sampling.

Usage

```
varepli(dat, nrep, g, np)
```

Arguments

`dat`	is a matrix containing in each row all the observed measurements along the time course experiment corresponding to each configuration. All replication measurements corresponding the same gene must be in consecutive rows. All genes must have the same number of replicates. Given n genes with R replicates and in a time course experiment with T time points, `dat` must be a (n*R) * T dimension matrix.
`nrep`	number of replications.
`g`	identification name for each gene.
`np`	number of the bootstrap sample.

Details

Each configuration is arranged in a two columns matrix where in the first column are the time points 1, ..., T and in the second column the observed measurements X=(x1, ..., xT).

To calculate the variability between R replicates of one object, the following procedure is carried out:

Step1: for each single gene and for each time point, there are R different observed measurements. Then, for each time point, one of them with replacement is drawn, defining a new replicate.

Step2: calculate the absolute value of the difference between the generalized procrustes statistic (see `prock`) related to the original replicates and the one related to the new resampling replicates generated in *Step1*.

Step3: repeat *Step1* and *Step2* np times in order to obtain the bootstrap distribution for the absolute value of the difference commented in the previous step. The variance of this distribution is a measure of the variability between the original replicates of the gene.

Value

It is a list with components:

`nsample`	bootstrap sample size.
`variability`	two column matrix with the modified generalized procrustes statistics values in the first column and the variance of the bootstrap distribution, in decreasing order, in the second.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Gower, J. C. and Dijksterhuis, G. B. (2004) *Procrustes Problems.* Oxford University Press.

See Also

`prock`, `serieprock`

Examples

```
# Generate 10 hypothetic time course profiles with T=12 time points, with 3 replicates.
T <- 12
R <- 3
x <- matrix(0, nrow=10*R, ncol=T)
for (i in 1: 10){
    m <- sample(1:10, T, replace=TRUE)
    x[(R*(i-1))+ 1,] <- m
    for (r in 2:R){
        x[(R*(i-1))+ r,] <- m + runif(T, -0.5, 0.5)
    }
}

id <- LETTERS[1:10] # identification of genes

varepli(dat=x, nrep=3,g=id, np=100)
```

vgeo *Geometric Variability*

Description

vgeo calculates the geometric variability for each group in data.

Usage

```
vgeo(d, pert = "onegroup")
```

Arguments

d	distance matrix or a dist object for the n individuals.
pert	vector of length n indicating which group each individual belongs to. The default value indicates there is only one group in data.

Value

It is a matrix containing the geometric variability for each group.

Author(s)

Itziar Irigoien ⟨itziar.irigoien@ehu.es⟩; Konputazio Zientziak eta Adimen Artifiziala, Euskal Herriko Unibertsitatea (UPV-EHU), Donostia, Spain.

Conchita Arenas ⟨carenas@ub.edu⟩; Departament d'Estadistica, Universitat de Barcelona, Barcelona, Spain.

References

Arenas, C. and Cuadras, C.M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, **2**, 183–191.

Cuadras, C.M. (1992). Some examples of distance based discrimination. *Biometrical Letters*, **29**, 3–20.

See Also

deltas, proxi

Examples

```
data(iris)
d <- dist(iris[,1:4])
vgeo(d,iris[,5])
```

Anexo B

Códigos fuente del paquete «DBmethods» en R

A continuación se adjuntan los códigos fuente de las funciones que forman el paquete «DBmethods» dentro del software de libre distribución R.

```
buscar_sol <- function(v_try, d, rango, varmin, varmax, var){
##############################################################################
#     Indicates whether there is a factible solution for
#            lambda=v_try and alpha=rango or not.
# Input:
# d: distance matrix.
# v_try: lambda.
# rango: alpha.
# varmin: minumum variability between replicates.
# varmax: maximum variability between replicates.
# var: vector of variabilities between replicates for each gene.
#
# Output:
# romper: when romper=1, there IS a factible solution; when romper=0, there
#         IS NOT a factible solution.
# lv_try: lambda
##############################################################################

n <- dim(d)[1]
eps <- 1.E-4

romper <- 1

label <- rep(0,n)   # indicates whether each object has been assigned
l <- rep(0,n)       # indicates the cluster it has been assigned

siguiente <- rep(0,n)  # indicates which object has been assigned after itself.
                       # siguiente[i]=j means after object i, there has been
                       # assigned objetc j.
val <- rep(1.E8, n)

i_cluster <- 1    # indicates the cluster will be assigned
indice <- 1
label[indice] <- 1
l[indice] <- i_cluster
actual <- indice
ultimo <- actual
siguiente[actual] <- 0


# Variabilitues between replicates are ranged in e intervals
if (var[indice]<= varmin+rango+eps ){
    if (var[indice]>= varmax-rango-eps){ intervalo <- 0
    } else {intervalo <- 1}
} else { intervalo <- 2}



while (actual != 0){
    for(i in 1:n){
```

```
      if (i != actual){
        cond <- condicion(valor=var[i], intervalo, varmax=varmax,
                          varmin=varmin, rango=rango)
            if ((d[actual,i] <= v_try+eps)&(cond)&(label[i]==0)){
               label[i] <- 1
               l[i] <- i_cluster
               siguiente[ultimo] <- i
               ultimo <- i
               if (intervalo==0){
                  if (var[i] <= varmax-rango-eps){intervalo <- 1
                  }  else {
                  if (var[i] >= varmin + rango + eps) {intervalo <- 2}
                  }
               }
            }
          if ((label[i]==1)& (l[i]==i_cluster)&
              (d[actual,i] <= val[i])){val[i] <- d[actual, i]
          }
       }
    }
    actual <- siguiente[actual]
}



indice <- 2
while ((indice <=n) & label[indice]==1){
    indice <- indice +1
}



if (indice > n){
    romper <- 1      # could be anyone
    l[n] <- 2
}

if (indice <=n){
    i_cluster <- 2
    label[indice] <- 1
    l[indice] <- i_cluster
    # val[indice] <- 0
    actual <- indice
    ultimo <- actual
    siguiente[actual] <- 0
    if (var[indice]<= varmin+rango+eps ){
        if ((var[indice]>= varmax-rango-eps)){ intervalo <- 0
        } else {intervalo <- 1}
    } else { intervalo <- 2
    }
```

```
    while (actual != 0){
        for(i in 1:n){
          if (i != actual){
            cond <- condicion(valor=var[i], intervalo,
                              varmax=varmax, varmin=varmin, rango=rango)
            if ((d[actual,i] <= v_try+eps) &(cond)& (label[i]==0)){
                  label[i] <- 1
                  l[i] <- i_cluster
                  siguiente[ultimo] <- i
                  ultimo <- i
                  if (intervalo==0){
                    if (var[i] <= varmax-rango-eps){intervalo <- 1
                    }  else {
                      if (var[i] >= varmin + rango + eps) {intervalo <- 2}
                    }
                  }
            }
            if ((label[i]==1)& (l[i]==i_cluster)&
                (d[actual,i] <= val[i])){val[i] <- d[actual, i]}
          }
        }
        actual <- siguiente[actual]
    } # while (actual != 0){
}

if (sum(label)<n){
    romper <- 0
}


out <- list(v_try=v_try, romper=romper, l=l)

return(out)
}
```

```
buscar_solSINreplicas <- function(v_try, d){
##############################################################################
#  Indicates whether there is a factible solution for  lambda=v_try or not.
# Input:
# d: distance matrix
# v_try: lambda
# Output:
#  romper: when romper=1, there IS a factible solution; when romper=0, there
#         IS NOT a factible solution.
# l: indicates the partition when it exits.
##############################################################################

n <- dim(d)[1]
eps <- 1.E-8

romper <- 1

label <- rep(0,n) # indicates whether each object has been assigned
l <- rep(0,n)     # indicates the cluster it has been assigned.

siguiente <- rep(0,n)  # indicates which object has been assigned after itself.
                       # siguiente[i]=j means after object i, there has been
                       # assigned objetc j.


val <- rep(1.E8, n)

indice <- 1
label[indice] <- 1
l[indice] <- 1
actual <- indice
ultimo <- actual
siguiente[actual] <- 0


while (actual != 0){
    for(i in 1:n){
      if (i != actual){
        if ((d[actual,i] <= v_try-eps)&(label[i]==0)){
            label[i] <- 1
            l[i] <- 1
            siguiente[ultimo] <- i
            ultimo <- i
            siguiente[i] <- 0
        }
      }
    }
    actual <- siguiente[actual]
}

indice <- 2
while ((indice <=n) & label[indice]==1){
    indice <- indice +1
```

```
}

if (indice > n){
    romper <- 1           # could be anyone
    l[n] <- 2
}


if (indice <=n){
   label[indice] <- 1
   l[indice] <- 2
   actual <- indice
   ultimo <- actual
   siguiente[actual] <- 0
   while (actual != 0){
        for(i in 1:n){
             if (i != actual){
                 if ((d[actual,i] <= v_try+eps) & (label[i]==0)){
                    label[i] <- 1
                    l[i] <- 2
                    siguiente[ultimo] <- i
                    ultimo <- i
                 }
             }
        }
        actual <- siguiente[actual]
   } # while (actual != 0){
}


for (i in 1:n){
    if (label[i]==0){romper <- 0}
}


out <- list(v_try=v_try, romper=romper, l=l)
}
```

```
center <- function(dat){
############### Centers and standardizes the configurations  ##################
#      such that trace(X'i*Xi)=1,  before the procrustes analysis
# Input:
# dat: array with configurations
# Output:
# dat: centered and standardized data.
################################################################################

nrep<-dim(dat)[3]
tpoints<-dim(dat)[1]
p<-dim(dat)[2]
datini<-dat

if (is.na(nrep)){  #When no a third dimension, theres only one configuration
   nrep <- 1
   dat<- array(dat, dim=c(tpoints,p,1))
}


# Centering matrix
one=rep(1,tpoints)
J=one %*% t(one)
I=diag(tpoints) #identity matrix
H=I-(1/tpoints)*J #centering matrix

# Centered configurations
for (i in 1:nrep){
dat[,,i]<- H %*% dat[,,i]
}


# Standarize data between different configurations
# Trace (Xi' Xi)=trace(Xj' Xj)=1

for (i in 1:nrep){
A<-0
  for (j in 1:p){
     A <- A + var(dat[,j,i])
  }

a<-sqrt((tpoints-1)*A)
dat[,,i] <- dat[,,i]/a
}

if (is.na(dim(datini)[3])){
dat<-dat[,,1]
}

return(dat)
}
```

```
CLAPPER <- function(d, renom, replicates=FALSE, DV=NULL, coef=2/3){
############### CLAPPER: calculates a division in two clusters  ##################
# Input:
# d: distance matrix
# renom: object names
# replicates: logical value indicating whether there are replications of the
#             experiment or not.
# DV: in case there is replication, the square roots of the geometric
#     variability between replicates
# coef: coefficient to stablish the alpha value of the algorithm.
#
# Output:
# n1: number objects in C1.
# g1: objects in C1.
# dp1: distance path of cluster C1.
# n2: number objects in C2.
# g2: objects in C2.
# dp2: distance path of cluster C2.
##################################################################################


if (replicates){
    if (is.null(DV)){
        stop("There are not values for variability between replicates")
    }
    out <- clapperCONreplicas(d, renom, DV, coef)
} else {
    out <- clapperSINreplicas(d, renom)
}


return(out)
}
```

```
clapperCONreplicas <- function(d, nombres, DV, coeff=2/3){
############## CLAPPER WITH REPLICATES  ######################################
#         calculates a division in two clusters
# Input:
# d: distance matrix
# nombres: object names
# DV: in case there is replication, the square roots of the geometric
#     variability between replicates.
# coeff: coefficient to stablish the alpha value of the algorithm.
#
# Output:
# n1: number objects in C1.
# g1: objects in C1.
# pd1: distance path of cluster C1.
# n2: number objects in C2.
# g2: objects in C2.
# pd2: distance path of cluster C2.
##############################################################################

eps <- 1E-8

d <- as.matrix(d)

n <- dim(d)[1] # number of inital genes.

var <- DV  # var. between replicates for each gene.

l <- rep(0,n)      # indicates the cluster it has been assigned.
lfin <- rep(0,n)   #indicates the cluster it has been assigned, LAST assignment

# Calculate alpha of the algorithm(rango)
varmin <- min(var)
varmax <- max(var)
rango <- coeff*(max(var)-min(var))

vmin <- 0
vmax <- 0

orden2 <-matrix(0,nrow=n, ncol=n)
ordent <- rep(0, n^2)
at <- rep(0, n^2)


for (i in 1:n){
    orden <- 1:n
    a <- d[i,]
    orden <- order(a)
    for (j in 1:n){
        orden2[i,j] <- orden[j]
        ordent[n*(i-1)+j] <- n*(i-1)+j
        at[n*(i-1)+j] <- d[i,j]
    }
    if (a[orden2[i,2]] > vmin+eps){
```

```
        vmin <- a[orden2[i,2]]
        i_min <- i
        index_min <- n*(i-1) + 1
    }
}



d_max <- 0
for (i in 1:n){
    for (j in 1:n){
        if ((i != i_min)&(j != i_min)&(d[i,j] > d_max)){d_max <- d[i,j]}
    }
}

g1 <- rep(0,n)
g2 <- rep(0,n)
if (d_max < vmin){
    v_sol <- d_max
    n1 <- 1
    g1[n1] <- nombres[i_min]
    l[i_min] <- 1
    vs1 <- 0
    n2 <- 0
    vs2 <- d_max
    for (i in 1:n){
        if (i != i_min){
            n2 <- n2+1
            g2[n2] <- nombres[i]
            l[i] <- 2
        }
    }
    lfin <- l
}  else{ # Binary search
    ordent <- order(at)
    index_min <- 1
    index_max <- n*n
    while ((at[ordent[index_min+1]] <= vmin+eps)&(index_min < index_max)){
        index_min <- index_min+1
    }
    vmin <- at[ordent[index_min]]
    vmax <- at[ordent[n*n]]
    while ((abs(at[ordent[index_max-1]] - vmax) <= eps)&
           (index_max > index_min)){index_max <- index_max -1}
    vmax <- at[ordent[index_max]]
    vs1 <- vmax
    vs2 <- vmax

    # Begin the binary search
    while (vmin < vmax){
        v_try <- vmin + (vmax-vmin)/2
        index_try <- index_min
        while (at[ordent[index_try+1]] <= v_try +eps){
```

```
            index_try <- index_try+1
        }
        v_try <- at[ordent[index_try]]

        romper <- 0


        aux <- buscar_sol(v_try, d, rango, varmin, varmax, var)
        v_try <- aux$v_try
        romper <- aux$romper
        l <- aux$l
        v1 <- aux$v1
        v2 <- aux$v2

        if (romper==1){
            index_max <- index_try
            v_max <- v_try
            while ((abs(at[ordent[index_max-1]]-v_max) <= eps)&
                   (index_max > index_min)){index_max <- index_max-1}
            vmax <- at[ordent[index_max]]
            lfin <- l
            v_sol <- v_try
            vs1 <- v1
            vs2 <- v2
        } else{
            index_min <- index_try+1
            vmin <- at[ordent[index_min]]
              while (at[ordent[index_min+1]] <= vmin +eps){
                 index_min <- index_min + 1
              }
        }
    } # while (vmin < vmax){
} # if (d_max < vmin){

    n1 <- 0
    n2 <- 0
    for (i in 1:n){
        if (lfin[i]==1){
            n1 <- n1 +1
            g1[n1] <- nombres[i]
        } else{ if (lfin[i]==2){
                n2 <- n2 +1
                g2[n2] <- nombres[i]
              }
        }
    }

# Distance matrices for each cluster C1 and C2
count11 <- 0
count21 <- 0
d1 <- matrix(0, nrow=n1, ncol=n1)
d2 <- matrix(0, nrow=n2, ncol=n2)
for (i in 1:n){
```

```
    count12 <- 0
    count22 <- 0
    if (lfin[i]==1){
        count11 <- count11+1
        for (j in 1:n){
            if (lfin[j]==1){
                count12 <- count12+1
                d1[count11,count12] <- d[i,j]
            }
        }
    } else{
      if (lfin[i]==2){
        count21 <- count21+1
        for (j in 1:n){
            if (lfin[j]==2){
                count22 <- count22+1
                d2[count21,count22] <- d[i,j]
            }
        }# for (j in 1:n){
      }
    }
}


pd1 <- path.d(d1)

pd2 <- path.d(d2)


out <- list(n1=n1, g1=g1[1:n1], pd1=pd1, n2=n2, g2=g2[1:n2], pd2=pd2)


return(out)

} # function
```

```
clapperSINreplicas <- function(d, nombres){
############### CLAPPER WITHOUT REPLICATES ###################################
#         calculates a division in two clusters
# Input:
# d: distance matrix
# nombres: object names
#
# Output:
# n1: number objects in C1.
# g1: objects in C1.
# pd1: distance path of cluster C1.
# n2: number objects in C2.
# g2: objects in C2.
# pd2: distance path of cluster C2.
##############################################################################

d <- as.matrix(d)
eps <- 1E-8

n <- dim(d)[1] # number of initial objects.


l <- rep(0,n)    # indicates the cluster it has been assigned.
lfin <- rep(0,n) #indicates the cluster it has been assigned, LAST assignment


vmin <- 0
vmax <- 0

orden2 <-matrix(0,nrow=n, ncol=n)
ordent <- rep(0, n^2)
at <- rep(0, n^2)


for (i in 1:n){
    orden <- 1:n
    a <- d[i,]
    orden <- order(a)
    for (j in 1:n){
        orden2[i,j] <- orden[j]
        ordent[n*(i-1)+j] <- n*(i-1)+j
        at[n*(i-1)+j] <- d[i,j]
    }
    if (a[orden2[i,2]] > vmin+eps){
        vmin <- a[orden2[i,2]]
        i_min <- i
        index_min <- n*(i-1) + 1
    }
}

d_max <- 0
for (i in 1:n){
    for (j in 1:n){
```

```
        if ((i != i_min)&(j != i_min)&(d[i,j] > d_max)){d_max <- d[i,j]}
    }
}



g1 <- rep(0,n)
g2 <- rep(0,n)
if (d_max < vmin){
    v_sol <- d_max
    n1 <- 1
    g1[n1] <- nombres[i_min]
    l[i_min] <- 1
    vs1 <- 0
    n2 <- 0
    vs2 <- d_max
    for (i in 1:n){
        if (i != i_min){
            n2 <- n2+1
            g2[n2] <- nombres[i]
            l[i] <- 2
        }
    }
    lfin <- l
}  else{ # Binary search
    ordent <- order(at)
    index_min <- 1
    index_max <- n*n
    while ((at[ordent[index_min+1]] <= vmin+eps)&(index_min < index_max)){
        index_min <- index_min+1
    }
    vmin <- at[ordent[index_min]]
    vmax <- at[ordent[n*n]]
    while ((abs(at[ordent[index_max-1]] - vmax) <= eps)&
           (index_max > index_min)){index_max <- index_max -1}
    vmax <- at[ordent[index_max]]
#    vs1 <- vmax
#    vs2 <- vmax

    # Begin the search
    while (vmin < vmax){
        v_try <- vmin + (vmax-vmin)/2
        index_try <- index_min
        while (at[ordent[index_try+1]] <= v_try +eps){
            index_try <- index_try+1
        }
        v_try <- at[ordent[index_try]]

        romper <- 0


        aux <- buscar_solSINreplicas(v_try, d)
        v_try <- aux$v_try
```

```
        romper <- aux$romper
        l <- aux$l

        if (romper==1){
            index_max <- index_try
            v_max <- v_try
            while ((abs(at[ordent[index_max-1]]-v_max) <= eps)&
                   (index_max > index_min)){index_max <- index_max-1}
            vmax <- at[ordent[index_max]]
            lfin <- l
            v_sol <- v_try
        } else{


            index_min <- index_try+1
            vmin <- at[ordent[index_min]]
              while (at[ordent[index_min+1]] <= vmin +eps){
                 index_min <- index_min + 1
              }
        }
    } # while (vmin < vmax){
} # if (d_max < vmin){

n1 <- 0
n2 <- 0
for (i in 1:n){
    if (lfin[i]==1){
        n1 <- n1 +1
        g1[n1] <- nombres[i]
    } else{ if (lfin[i]==2){
            n2 <- n2 +1
            g2[n2] <- nombres[i]
          }
    }
}

# Distance matrices of clusters C1 and C2
count11 <- 0
count21 <- 0
d1 <- matrix(0, nrow=n1, ncol=n1)
d2 <- matrix(0, nrow=n2, ncol=n2)
for (i in 1:n){
    count12 <- 0
    count22 <- 0
    if (lfin[i]==1){
        count11 <- count11+1
        for (j in 1:n){
            if (lfin[j]==1){
                count12 <- count12+1
                d1[count11,count12] <- d[i,j]
            }
        }
    } else{
```

```
     if (lfin[i]==2){
        count21 <- count21+1
        for (j in 1:n){
            if (lfin[j]==2){
                count22 <- count22+1
                d2[count21,count22] <- d[i,j]
            }
        } # for (j in 1:n){
     }
   }
}


pd1 <- path.d(d1)
pd2 <- path.d(d2)

out <- list( n1=n1, g1=g1[1:n1],  pd1=pd1, n2=n2, g2=g2[1:n2],  pd2=pd2)
return(out)
} # function
```

```
condicion <- function(valor, intervalo, varmax, varmin, rango){
################ Variability between replicates ################
# Depending the var. between replicates the object has, evaluates whether
# the object can be assigned to the considered cluster.
# Input:
# valor: variability between replicates of the object.
# intervalo: indicates the kind of var between replicates for the already
#            assigned objects.
# varmax: maximum var. between replicates.
# varmin: manimum var. between replicates.
# rango: alpha value of the algorithm.
# Output:
# cond: logical.
#################################################################################
eps <- 1.E-8

if (intervalo==0){
cond <- TRUE
}
else {
     if ((intervalo ==1) & (valor <= varmin+rango+eps)){cond <- TRUE}
     if ((intervalo ==2) & (valor >= varmax - rango - eps)){cond <- TRUE}
     if ((intervalo ==2) & (valor <= varmax - rango - eps)){cond <- FALSE}
     if ((intervalo ==1) & (valor >= varmin+rango+eps)){cond <- FALSE}
}
out <- cond
return(out)
}
```

```
dbhatta <- function(x){
########## Bhattacharyya distance ######################
# Input:
# x: data matrix
# Output:
# d: bhattacharyya distance matrix
#########################################################
x <- as.matrix(x)
n <- dim(x)[1]
p <- dim(x)[2]
d <- matrix(0, nrow=n, ncol=n)
for (i in 1:(n-1)){
for (j in (i+1):n){
a <- sqrt(x[i,]*x[j,])
a <- sum(a)
d[i,j] <- acos(a)
d[j,i] <- d[i,j]
}
}
return(d)
}
```

```
dcor<- function(x){
########## Correlation distance between pais of objects ##################
# Input:
# x: data matrix
#
# Output:
# d: distance matrix
##########################################################################
    n<- dim(x)[1]
    p <- dim(x)[2]

    d<- matrix(0, n,n)
    for (i in 1:(n-1)){
         for (j in (i+1):(n-1)){
            r<- cor(x[i,],x[j,])
            d[i,j] <- sqrt(1-r)
            d[j,i] <- d[i,j]
         }
     }


return(d)
}
```

```
deltas <-function(d,pert="onegroup"){
############ Calculates distances between groups ######################
# Input:
# d: distance matrix between individuals (nxn)
# pert: integer vector indicating the group each individual belongs to.
# Output:
# delta: kxk matrix with distances between groups
#########################################################################

d <- as.matrix(d)
n<-dim(d)[1]
if (pert[1]=="onegroup"){pert <- rep(1,n)}
pert <- as.integer(pert)
k<-max(pert)

# We need the geometrical variabilities
var <- vgeo(d,pert)

# We need the squared distances
d <- d*d


delta <- matrix(0, k,k) #matrix of distances between groups

frec <- matrix(0,k,1) # vector of frecuencies of individuals in each population

for (i in 1:n){
  for (pob in 1:k){
      if (pert[i]==pob){
          frec[pob]<- frec[pob]+1
      }
  }
}


for (i in 1:n){
   for (j in 1:n){
       if (pert[i]!=pert[j]){
          if (pert[i]<pert[j]){
             delta[pert[i],pert[j]]=delta[pert[i],pert[j]]+d[i,j]
          }

          if (pert[i]>pert[j]){
             delta[pert[j],pert[i]]=delta[pert[j],pert[i]]+d[i,j]
          }


       }

   }
}
```

```
for (pob1 in 1:k){
    for (pob2 in pob1:k){
        delta[pob1, pob2]=delta[pob1, pob2]/
                          (2*frec[pob1]*frec[pob2])-var[pob1]-var[pob2]

    }
}

for (pob1 in 1:k){
    for (pob2 in pob1:k){
        delta[pob2, pob1]=delta[pob1, pob2]
    }
}

for (pob in 1:k){delta[pob,pob]=0}


return(delta)
}
```

```
deltas_simple <-function(d,var,pert){
############ Distances between groups (faster version) ##########################
# Input:
# d: distance matrix(nxn)
# var: vector of geometric variabilities
# pert:  integer vector indicating the group each individual belongs to.
# Output:
# delta: kxk matrix with distances between populations
#################################################################################

d <- d*d

n<-dim(d)[1]
k<-max(pert)
delta <- matrix(0, k,k)

frec <- matrix(0,k,1) # vector of frecuencies of individuals in each population

for (i in 1:n){
  for (pob in 1:k){
      if (pert[i]==pob){
          frec[pob]<- frec[pob]+1
      }
  }
}


for (i in 1:n){
   for (j in 1:n){
       if (pert[i]!=pert[j]){
          if (pert[i]<pert[j]){
             delta[pert[i],pert[j]]=delta[pert[i],pert[j]]+d[i,j]
          }

          if (pert[j]>pert[i]){
             delta[pert[j],pert[i]]=delta[pert[j],pert[i]]+d[i,j]
          }


       }

   }
}


for (pob1 in 1:k){
    for (pob2 in pob1:k){
        delta[pob1, pob2]=delta[pob1, pob2]/
                          (2*frec[pob1]*frec[pob2])-var[pob1]-var[pob2]

    }
}
```

```
for (pob1 in 1:k){
    for (pob2 in pob1:k){
        delta[pob2, pob1]=delta[pob1, pob2]
    }
}

for (pob in 1:k){delta[pob,pob]=0}


return(delta)
}
```

```
dgower <- function(x, type=list()){
############ Gower distance for mixed data ###############
# Input:
# x: data matrix. Missing values are indicated as "NA"
# type: list with components,
#      cuant =  position of cuantitative variables
#      bin = position of  binary variables (asymmetric)
#      nom = position of nominal variables
# Output: d: distance matrix
##########################################################
dx <- dim(x)
n <- dx[1]
p <- dx[2]
ntype <- names(type)   # given names in type

if (length(type)){
    if ((!is.list(type)) || (is.null(ntype)) || any(ntype == "")){
        stop("invalid ", sQuote("type"), "; must be named list")
    }
   for (nt in ntype){
        cvec <- type[[nt]]
        if (is.numeric(cvec)){
            if (!all(1 <= cvec & cvec <=p)){
                 stop("type$", nt, " must be in 1:ncol(x)")
            }
        } else {  stop("type$", nt, " must  contain numbers")}
    }
    vc <- type$cuant
    vb <- type$bin
    vn <- type$nom
    v <- c(vc, vb, vn)
    if (!all(sort(v) == 1:p)){
       stop("type must contain specifications for all variables")
    }
} else { stop("type must contain specifications for all variables")}
# Now we got all variables types in vc, vb and vn

# Are there missing values??
if (any(is.na(x))){
    missing <- TRUE
} else{
    missing <- FALSE
}

if (missing){
    d <- gowerWITHmissing(x, vc, vb, vn)
} else{
    d <- gowerNOmissing(x, vc, vb, vn)
}
return(d)
}
```

```
distrW <- function(d, pert, np, frec, frecacum){
################## Bootstrap distribution of W under H0 ######################
# Input:
# d: Distance matrix
# pert: integer vector indicating the group each individual belongs to.
# np: sample size for the bootstrap distribution
# Output:
# TW: null distribution of W
##############################################################################

n<-dim(d)[1] # there are n elements
k<-max(pert) # there are k populations

dboot <- matrix(0,n,n)
TW <- matrix(0, np,1)+999

indor<-order(pert,c(1:n)) # <- Names of INDIVIDUALS ordered by pert

    for (l in 1:np){
        #choose one ind. known to be from one initial group, and calculate its W
        indatestar=round(runif(1,0.5,n+0.5))

        indelegidos <- matrix(0, 1,n)
         # Obtain sample with size n_pob (frec[pob]) from the original groups
         # They are in the vector "indelegidos"
        for (pob in 1:k){
            for (i in 1:frec[pob]){
                cual <- round(runif(1,0.5, frec[pob]+0.5))
                indelegidos[frecacum[pob]+i] <- indor[frecacum[pob]+cual]
            }
        }

        # Construct the new distance matrix
        for (i in 1:frecacum[k+1]){
            for (j in 1:frecacum[k+1]){
                dboot[i,j]=d[indelegidos[i],indelegidos[j]]
            }
        }


        # Calculate geom. var. and deltas
        # pert keeps the cluster each generated ind. belongs to.
        vg <- vgeo(dboot, pert)
        delta<-deltas_simple(dboot,vg, pert)

        #calculate distances from indatestar to clusters
        dx0b <- matrix(0, 1,n)

        for (i in 1:n){
            dx0b[i]=d[indatestar, indelegidos[i]];
        }

        phib <- proxi_simple(dx0b,vg, pert)
```

```
        TW[l] <- estW_simple(vg,delta,phib)$Wvalue

    } # for l in 1:np


return(TW)
}
```

```
dmahal<- function(datos, S){
########## Mahalanobis distance between pais of objects ##################
# Input:
# datos: data matrix
# S: var-cov matrix
# Output:
# d: distance matrix
#############################################################################
    n<- dim(datos)[1]
    p <- dim(datos)[2]
    s1 <- dim(S)[1]
    s2 <- dim(S)[2]
    if (s1 != s2){stop("S must be squared matrix")}
    if(p != s1){stop("data and covariance matrix must have coherent dimesions")}
    d<- matrix(0, n,n)
    I <- diag(rep(1,p)) #identity matrix
    Sinv <- solve(S, I)
    for (i in 1:n){
         for (j in 1:i){
            a<- datos[i,]-datos[j,]
            a<-matrix(a, nrow=2,ncol=1)
            d[i,j] <- sqrt(t(a)%*%Sinv%*%a)
         }
     }
    for (i in 1:n){
         for (j in 1:i){
            d[j,i] <- d[i,j]
         }
     }


return(d)
}
```

```
estW<-function(d, dx0, pert="onegroup"){
################## ESTIMATION OF INCA STATISTIC ###################################
# Input:
# d: distance matrix between individuals (nxn)
# dx0: vector of length n with distancies from the specific individual to the
#      individuals of different groups.
# pert: integer vector indicating the group each individual belongs to.
#
# Output: value of INCA statistic (W), and values of projections (U1, ..., Uk)
###################################################################################

d <- as.matrix(d)
n<-dim(d)[1]
if (pert[1]=="onegroup"){pert <- rep(1,n)}
pert <- as.integer(pert)
k<-max(pert)

# We need the geometrical variabilities
var <- vgeo(d,pert)

delta <- deltas(d, pert)

phi <- proxi(d, dx0, pert)


W <-0
w1<-0
w2<-0
w3<-0

alpha <- matrix(0,k-1,1)
U<-matrix(0,k,1)
M<- matrix(0,k-1,k-1)
N<- matrix(0,k-1,1)


for (i in 1:(k-1)){
    for (j in i:(k-1)){

        M[i,j]=delta[i,k]+delta[j,k]-delta[i,j]

    }
}

for (i in 1:(k-1)){
    for (j in i:(k-1)){
        M[j,i]=M[i,j]
    }
}

for (i in 1:(k-1)){
    N[i]=delta[i,k]+phi[k]-phi[i]
}
```

```
alpha<-ginv(M)%*%N

for (i in 1:(k-1)){
    w1<- w1+alpha[i]*(delta[i,k]+phi[k]-phi[i])
}


for (i in 1:(k-1)){
    w2=w2+alpha[i]*alpha[i]*delta[i,k]
}

for (i in 1:(k-1)){
  if (i < k-1){
    for (j in (i+1):(k-1)){
      w3=w3+alpha[i]*alpha[j]*(delta[i,k]+delta[j,k]-delta[i,j]);
    }
  }
}

W<- phi[k]-w1+w2+w3;



if (W<0){W<-0}

for (i in 1:k){
    U[i]=phi[i]-W
}
out=list(Wvalue=W,  Uvalue=U)

return(out)
}
```

```
estW_simple<-function(var, delta, phi){
######################### Estimation of INCA W (faster version) ###############
# Input:
# var: vector of geometric variabilities
# delta: matrix of distances between clusters
# phi: proximity vector
# Output:
# Wvalue: estimation of W
# Uvalue: projections U1, ..., Uk
###############################################################################
k <- length(var)
alpha <- matrix(0,k-1,1)
U<-matrix(0,k,1)
M<- matrix(0,k-1,k-1)
N<- matrix(0,k-1,1)

for (i in 1:(k-1)){
    for (j in i:(k-1)){
        M[i,j]=delta[i,k]+delta[j,k]-delta[i,j]
    }
}
for (i in 1:(k-1)){
    for (j in i:(k-1)){
        M[j,i]=M[i,j]
    }
}
for (i in 1:(k-1)){
    N[i]=delta[i,k]+phi[k]-phi[i]
}
alpha<-ginv(M)%*%N
alpha <- c(alpha, 1-sum(alpha))
W1 <- 0
for (i in 1:k){
        W1 <- W1 + alpha[i]*phi[i]
}
W2 <- 0
for (i in 1:(k-1)){
        for (j in (i+1):k){
                W2 <- W2 + alpha[i]*alpha[j]*delta[i,j]
        }
}
W <- W1-W2
if ( W < 0 ){W <- 0}
for (i in 1:k){
    U[i]=phi[i]-W
}
out=list(Wvalue=W, Uvalue=U)
return(out)
}
```

```
getnns <- function(diss, flag) {
###################### Nearest neighbor ####################################
# Input:
# diss: full distance matrix.
# flag: "live" rows indicated by 1 are to be processed.
#
# Output:
# List of: nn, nndiss.
# nn:   list of nearest neighbor of each row.
# nndiss: nearest neighbor distance of each row.
# FM, 2003/11/16
############################################################################

   nn <- rep(0, nrow(diss))
   nndiss <- rep(0.0, nrow(diss))
   MAXVAL <- 1.0e12
   if (nrow(diss) != ncol(diss)) stop("Invalid input first parameter.")
   if (nrow(diss) != length(flag)) stop("Invalid inputs 1st/2nd parameters.")
 # if (nrow(diss) != length(nn)) stop("Invalid inputs 1st/3rd parameters.")
 # if (nrow(diss) != length(nndiss)) stop("Invalid inputs 1st/4th parameters.")

   for (i1 in 1:nrow(diss)) {
       if (flag[i1] == 1) {
          minobs <- -1
          mindis <- MAXVAL
          for (i2 in 1:ncol(diss)) {
              if ( (diss[i1,i2] < mindis) && (i1 != i2) ) {
                 mindis <- diss[i1,i2]
                 minobs <- i2
              }
          }
          nn[i1] <- minobs
          nndiss[i1] <- mindis
       }
   }
   list(nn = nn, nndiss = nndiss)
}
```

```
GEVAdivisive <- function(d, renom=NULL){
################## Performs the GEVAdivisive partition of C in clusters C1 and C2
# Input:
# d: distance matrix
# renom: names of objects
#
# Output:
# VT: criterium value
# n1: size of the cluster C1
# g1: objects in C1
# n2: size of the cluster C2
# g1: objects in C2
#
# Note: needs combn() from library utils
#############################################################################

    d <- as.matrix(d)
    n <- dim(d)[1]
    if (is.null(renom)){
       renom <- 1:n
    }
    d2 <- d*d                     # Squared matrices to calculate the geo. var.
    nn <- floor(n/2)
    VTfin <- 1e12
    for (r in 1:nn){
        cb <- combn(n, r)
        for (i in 1:dim(cb)[2]){
              codi <- matrix(0, nrow=n, ncol=1)
              codic <- matrix(0, nrow=n, ncol=1)
              codi[cb[,i]] <- 1
              codic <- 1-codi
              sum1 <- t(codi)%*% d2 %*%codi
              sum1 <- sum1/(2*sum(codi))
              sum2 <- t(codic)%*% d2 %*%codic
              sum2 <- sum2/(2*sum(codic))
              VT <- sum1 + sum2
              if (VT < VTfin){
                   VTfin <- VT
                   codifin <- codi
              }
        }
    }

    codicfin <- 1 - codifin
    n1 <- sum(codifin)
    g1 <- rep(NA, n1)
    aux <- 0
    for (i in 1:n){
       if (codifin[i]==1){
           aux <- aux + 1
           g1[aux] <- renom[i]
       }
    }
```

```
    n2 <- sum(codicfin)
    g2 <- rep(NA, n2)
    aux <- 0
    for (i in 1:n){
         if (codicfin[i]==1){
             aux <- aux + 1
             g2[aux] <- renom[i]
         }
    }

out <- list(VT= VTfin, n1=n1, g1=g1, n2=n2, g2=g2)

return(out)
}
```

```
GEVAjoining <- function(d) {
####### GEVAjoining. Constructs the agglomerative process ##################
#            Based on the hierclust.R function by Mutargh
#
# Input:
# d: distance matrix.
# Output:
# A list of class "hclust"
#############################################################################

   MAXVAL <- 1.0e12
   d <- as.matrix(d)
   d2 <- d*d
   n <- nrow(d)
  # diss <- dissim(a, wt)                       # call to function dissim
   dorig <- d2
   diss <- d2/4  # initial geometric variability between pairs
   flag <- rep(1, n)                           # active/dead indicator
   a <- rep(0, n-1)                            # left subnode on clustering
   b <- rep(0, n-1)                            # right subnode on clustering
   ia <- rep(0, n-1)                           # R-compatible version of a
   ib <- rep(0, n-1)                           # R-compatible version of b
   lev <- rep(0, n-1)                          # level or criterion values
   card <- rep(1, n)                           # cardinalities
   #mass <- wt
   order <- rep(0, n)                          # R-compatible order for plotting

   nnsnnsdiss <- getnns(diss, flag)            # call to function getnns
   clusmat <- matrix(0, n, n)                  # cluster memberships
   for (i in 1:n) clusmat[i,n] <- i            # init. trivial partition

   for (ncl in (n-1):1) {                       # main loop
       # check for agglomerable pair
       minobs <- -1;
       mindis <- MAXVAL;
       for (i in 1:n) {
           if (flag[i] == 1) {
              if (nnsnnsdiss$nndiss[i] < mindis) {
                  mindis <- nnsnnsdiss$nndiss[i]
                  minobs <- i
              }
           }
       }
       # find agglomerands clus1 and clus2, with former < latter
       if (minobs < nnsnnsdiss$nn[minobs]) {
          clus1 <- minobs
          clus2 <- nnsnnsdiss$nn[minobs]
       }
       if (minobs > nnsnnsdiss$nn[minobs]) {
          clus2 <- minobs
          clus1 <- nnsnnsdiss$nn[minobs]
       }
       # So, agglomeration of pair clus1 < clus2 defines cluster ncl
```

```
       #------------------------------------ Block for subnode labels
       a[ncl] <- clus1                      # aine, or left child node
       b[ncl] <- clus2                      # benjamin, or right child node
       # Now build up ia, ib as version of a, b which is R-compliant
       if (card[clus1] == 1) ia[ncl] <- (-clus1)     # singleton
       if (card[clus2] == 1) ib[ncl] <- (-clus2)     # singleton
       if (card[clus1] > 1) {               # left child is non-singleton
          lastind <- 0
          for (i2 in (n-1):(ncl+1)) {       # Must have n-1 >= ncl+1 here
              if (a[i2] == clus1) lastind <- i2    # Only concerns a[i2]
          }
          ia[ncl] <- n - lastind            # label of non-singleton
       }
       if (card[clus2] > 1) {               # right child is non-singleton
          lastind <- 0
          for (i2 in (n-1):(ncl+1)) {       # Must have n-1 >= ncl+1 here
              if (a[i2] == clus2) lastind <- i2    # Can only concern a[i2]
          }
          ib[ncl] <- n - lastind            # label of non-singleton
       }
       if (ia[ncl] > 0 || ib[ncl] > 0) {    # Check that left < right
          left <- min(ia[ncl],ib[ncl])
          right <- max(ia[ncl],ib[ncl])
          ia[ncl] <- left                   # Just get left < right
          ib[ncl] <- right
       }
       #--------------------------------------------------------------------

       lev[ncl] <- mindis
       for (i in 1:n) {
           clusmat[i,ncl] <- clusmat[i,ncl+1]
           if (clusmat[i,ncl] == clus2) clusmat[i,ncl] <- clus1
       }
       # Next we need to update diss array
 elementos <- matrix(0,n,n)
 for (i in 1:n){
for (j in 1:n){
if (clusmat[i,ncl]==clusmat[j,ncl]){elementos[i,j] <- 1}
}
 }
       for (i in 1:n) {
           if ( (i != clus1) && (i != clus2) && (flag[i] == 1) ) {
  contar <- elementos[clus1,]+ elementos[i,]
  tot <- sum(contar)
              diss[clus1,i] <- t(contar) %*% dorig%*% contar /(2*tot*tot)
              diss[i,clus1] <- diss[clus1,i]
           }
       }
#       mass[clus1] <- mass[clus1] + mass[clus2]    # Update mass of new cluster
       card[clus1] <- card[clus1] + card[clus2]    # Update card of new cluster
       # Cluster label clus2 is knocked out; following not nec. but no harm
       flag[clus2] <- 0
```

```
        nnsnnsdiss$nndiss[clus2] <- MAXVAL
#        mass[clus2] <- 0.0
        for (i in 1:n) {
            diss[clus2,i] <- MAXVAL
            diss[i,clus2] <- diss[clus2,i]
        }
        # Finally update nnsnnsdiss$nn and nnsnnsdiss$nndiss
        # i.e. nearest neighbors and the nearest neigh. dissimilarity
        nnsnnsdiss <- getnns(diss, flag)
    }

    temp <- cbind(a,b)
    merge2 <- temp[nrow(temp):1, ]
    temp <- cbind(ia,ib)
    merge <- temp[nrow(temp):1,]
    dimnames(merge) <- NULL
    # merge is R-compliant; later suppress merge2

    #------------------------------- Build R-compatible order from ia, ib
    orderlist <- c(merge[n-1,1], merge[n-1,2])
    norderlist <- 2
    for (i in 1:(n-2)) {           # For precisely n-2 further node expansions
        for (i2 in 1:norderlist) {       # Scan orderlist
            if (orderlist[i2] > 0) {     # Non-singleton to be expanded
               tobeexp <- orderlist[i2]
               if (i2 == 1) {
                  orderlist <- c(merge[tobeexp,1],merge[tobeexp,2],
                                 orderlist[2:norderlist])
               }
               if (i2 == norderlist) {
                  orderlist <- c(orderlist[1:(norderlist-1)],
                                 merge[tobeexp,1],merge[tobeexp,2])
               }
               if (i2 > 1 && i2 < norderlist) {
                  orderlist <- c(orderlist[1:(i2-1)],
                                 merge[tobeexp,1],merge[tobeexp,2],
                                 orderlist[(i2+1):norderlist])
               }
               norderlist <- length(orderlist)
            }
         }
    }
    orderlist <- (-orderlist)
    class(orderlist) <- "integer"

    xcall <- "GEVAjoining(d)"
    class(xcall) <- "call"
    #clusmat=clusmat
    #labels=as.character(1:n)

    retlist <- list(merge=merge,height=as.single(lev[(n-1):1]),order=orderlist,
          labels=dimnames(a)[[1]],method="minvar",call=xcall,
          dist.method="euclidean-factor")
```

```
   retlist <- list(merge=merge,height=lev[(n-1):1],order=orderlist)
   class(retlist) <- "hclust"
   retlist
   return(retlist)
}
```

```
gowerNOmissing <- function(x, vc, vb, vn){
############### Calculates gower distance for mixed variables #######
##                  no missing values
# Input:
# x: data matrix
# vc: column position for cuantitative variables
# vb: column position for binary variables
# vn: column position for nominal variables
# Output:
# d: distance matrix
#########################################################################

x <- as.matrix(x)
dx <- dim(x)
n <- dx[1]
p <- dx[2]

d <- matrix(0, n, n)     # distance matrix
s <- matrix(0, n, n)      # similarity matrix

a <- matrix(0,n, n)      # concordance-concordance
di <- matrix(0,n, n)      # discordance-discordance

alfa <- matrix(0,n, n)      # concordance for nominal


R <- rep(0, p)

for (cont in vc){
    R[cont] <- max(x[,cont]) - min(x[, cont])
}

for (cont in vc){
    for (i in 1:(n-1)){
        for (j in (i+1):n){
             s[i,j] <- s[i,j] + (1 - abs(x[i, cont] - x[j, cont])/R[cont])
        }
    }
}


for (bin in vb){
    for (i in 1:(n-1)){
        for (j in (i+1):n){
             if (x[i, bin] == x[j, bin]){
                 if (x[i, bin] == 1){
                     a[i,j] <- a[i,j] + 1
                 } else {
                     di[i,j] <- di[i,j] + 1
                 }
             }
        }
    }
```

```
}

for (nom in vn){
    for (i in 1:(n-1)){
        for (j in (i+1):n){
             if (x[i, nom] == x[j, nom]){
                     alfa[i,j] <- alfa[i,j] + 1
             }
        }
    }
}

for (i in 1:(n-1)){
    for (j in (i+1):n){
       s[i,j] <- (s[i,j]+a[i,j]+alfa[i,j])/
                 (length(vc)+(length(vb)-di[i,j])+ length(vn))
    }
}


for (i in 1:(n-1)){
    for (j in (i+1):n){
       d[i,j] <- sqrt(2*(1 - s[i,j]))
       d[j,i] <- d[i,j]
    }
}

return(d)
}
```

```
gowerWITHmissing <- function(x, vc, vb, vn){
############### Calculates gower distance for mixed variables #######
##                  WITH missing values
# Input:
# x: data matrix
# vc: column position for cuantitative variables
# vb: column position for binary variables
# vn: column position for nominal variables
# Output:
# d: distance matrix
#####################################################################

x <- as.matrix(x)
dx <- dim(x)
n <- dx[1]
p <- dx[2]

w <- matrix(1, n, p)     # matrix of weights
for (i in 1:n){
    for (j in 1:p){
        if (is.na(x[i,j])){w[i,j] <- 0}
    }
}

d <- matrix(0, n, n)      # distance matrix
s <- matrix(0, n, n)      # similarity matrix

a <- matrix(0,n, n)       # concordance-concordance
di <- matrix(0,n, n)      # discordance-discordance

alfa <- matrix(0,n, n)    # concordance for nominal


R <- rep(0, p)

for (cont in vc){
    R[cont] <- max(x[,cont], na.rm = TRUE) - min(x[, cont], na.rm =TRUE)
}

for (i in 1:(n-1)){
     for (j in (i+1):n){
         swc <- 0                                # weights for cuant.
         swb <- 0                                # weights for bin.
         swn <- 0                                 # weights for nom.
         # Cuantitative variables
         for (cont in vc){
             if (!is.na(x[i, cont])){
                 if (!is.na(x[j, cont])){
                     swc <- swc + 1
                     s[i,j] <- s[i,j] + (1 - abs(x[i, cont] - x[j, cont])/R[cont])
                 }
             }
         }
```

```
         # Binary variables
         for (bin in vb){
             if (!is.na(x[i, bin])){
                  if (!is.na(x[j, bin])){
                       swb <- swb + 1
                       if (x[i, bin] == x[j, bin]){
                            if (x[i, bin] == 1){
                                a[i,j] <- a[i,j] + 1
                            } else {
                            di[i,j] <- di[i,j] + 1
                            }
                       }
                 }
             }
         }

          # Nominal variables
         for (nom in vn){
             if (!is.na(x[i,nom])){
                 if (!is.na(x[j,nom])){
                     swn <- swn + 1
                     if (x[i, nom] == x[j, nom]){
                         alfa[i,j] <- alfa[i,j] + 1
                     }
                 }
             }
         }


        s[i,j] <- (s[i,j] + a[i,j] + alfa[i,j])/(swc + (swb-di[i,j]) + swn)
        d[i,j] <- sqrt(2*(1-s[i,j]))
        d[j,i] <- d[i, j]

    } # for (j in (i+1):n){
} # for (i in 1:(n-1)){


return(d)
}
```

```
haysolucion <- function(v_try, d){
#############################################################################
#  Indicates whether there is a factible solution for  lambda=v_try
# Input:
# d: distance matrix
# var: vector with variabilities between replicates for each obj.
#
# Output:
# romper: when romper=1, there IS a factible solution; when romper=0, there
#         IS NOT a factible solution.
# l: indicates the partition when it exits.
# v1: path distance of cluster C1.
# g1: objects assigned to cluster C1.
# v2: path distance of cluster C2.
# g2: objects assigned to cluster C2.
#############################################################################
n <- dim(d)[1]
eps <- 1.E-8
romper <- 1
label <- rep(0,n) # indicates whether each object has been assigned
l <- rep(0,n)     # indicates the cluster it has been assigned
siguiente <- rep(0,n)  # indicates which object has been assigned after itself.
                       # siguiente[i]=j means after object i, there has been
                       # assigned object j.
val <- rep(1.E8, n)
i_cluster <- 1 # indicates cluster 1
indice <- 1
label[indice] <- 1
l[indice] <- i_cluster
# val[indice] <- 0
actual <- indice
ultimo <- actual
siguiente[actual] <- 0
while (actual != 0){
        for(i in 1:n){
          if (i != actual){
                if ((d[actual,i] <= v_try+eps)&(label[i]==0)){
                  label[i] <- 1
                  l[i] <- i_cluster
                  siguiente[ultimo] <- i
                  ultimo <- i
                  siguiente[i] <- 0
                }
          }
        }
        actual <- siguiente[actual]
}

if (sum(label) < n){romper <- 0}
out <- list(v_try=v_try, romper=romper)
return(out)
}
```

```
INCAnumclu <- function(d,pert_clus){
############### Number of atypical individuals  ##############################
# Input:
# d: distance matrix (n x n)
# pert_clus: partition of individuals  (n x 1)
# Output:
# Atypicals: Number of atipicals or well classified for each cluster given a
#            partition pert_clus
# Total : Mean percentage of atypicals(well classified) for the partition
#         pert_clus
# Ni_cluster: number of individuals in each cluster for the given partition
##############################################################################

d <- as.matrix(d)
n <- dim(d)[1]
pert_clus <- as.integer(pert_clus)
nclus<- max(pert_clus)
k <- nclus-1 # When there are 5 clusters, INCA is applied for 4 clusters
atypicals <- matrix(0,nclus,1)

percent_aty <- matrix(0, nclus,1)


f <- matrix(0, nclus,1) #frecuencies of each cluster
for (i in 1:n){
  for (pob in 1:nclus){
      if (pert_clus[i]==pob){
          f[pob]<- f[pob]+1
      }
  }
}

# Calculate W for ind. in each cluster and afterwards to compare with the rest

 for (tt in 1:nclus){    # se va a calcular INCA para el cluster tt
     # Calculate frecuencies in each cluster to apply INCA
     ff <- matrix(0,k,1)
     aux <-0
     for (i in 1:nclus){
        if (i !=tt){
           aux <- aux+1
           ff[aux]<-f[i]
        }
     }

     nn <- sum(ff)  # total  individual in INCA-typicality
     nv <- f[tt] # number of individual to be testes by INCA-typicality

     pert <- matrix(0, nn,1) # partition in k clusters
     aux <- 0
     for (i in 1:n){
        if (pert_clus[i] != tt){
           aux <- aux +1
```

```
          if (pert_clus[i]<=tt){
              pert[aux] <- pert_clus[i]
          }
         if (pert_clus[i]>tt){
              pert[aux] <- pert_clus[i]-1
          }
       }
     }



      xx_dist <- matrix(0,nn,nn)
      v_dist <- matrix(0,nv,nn)


      aux1 <- 0
      for (i in 1:(n-1)){
           if (pert_clus[i] != tt){
             aux1 <- aux1 +1
             aux2 <- aux1
             for (j in (i+1):n){
                if (pert_clus[j] != tt){
                   aux2 <- aux2+1
                   xx_dist[aux1, aux2] <- d[i,j]
                }
             }
         }
      }


     for (i in 1:nn){
         for (j in i:nn){
           xx_dist[j,i] <- xx_dist[i,j]
         }
     }
# Verified.

     aux1<-0
     for (i in 1:n){
        if (pert_clus[i]==tt){
           aux1<- aux1+1
           aux2 <-0
           for (j in 1:n){
              if (pert_clus[j]!=tt){
                  aux2<-aux2+1
                  v_dist[aux1,aux2] <- d[i,j]
              }
           }
        }
     }
```

```
     # Calculate INCA to each ind. in v_dist with respect data in xx_dist
        if (k==1){
           atypicals[tt] <- maxW_k1(xx_dist, v_dist)
           percent_aty[tt] <- atypicals[tt]/f[tt]
        }



       if (k>1){
           atypicals[tt] <- maxW_k(xx_dist, v_dist, pert)
           percent_aty[tt] <- atypicals[tt]/f[tt]
        }


   } # for (tt in 1:nclus)

total <- sum(percent_aty)/nclus


out <- list(well_class=atypicals, Ni_cluster=f, Total= total)

return(out)

} #end of function
```

```
INCAtest <- function(d, pert, d_test, np=1000, alpha=0.05, P=1){
#################### Application of INCA test ###################################
# Input:
# d: distance matrix between individuals (nxn)
# pert: integer vector indicating the group each individual belongs to.
# d_test: vector of length n with distancies from the specific individual to the
#         individuals of different groups.
# np: bootstrap sample size for null distribution of W
# alpha: fixed level of the test
# P: the procedure is repeted 10*P times
# output: value of INCA statistic (W), pvalue and values of proj. (U1,..., Uk)
#################################################################################
d <- as.matrix(d)
n<-dim(d)[1]
pert <- as.integer(pert)
k<-max(pert)

# Calculate  W and projections U
vg <- vgeo(d, pert);
delta <- deltas_simple(d,vg,pert)
phi <- proxi_simple(d_test,vg,pert)
ama <- estW_simple(vg,delta, phi)
W0 <-  ama$Wvalue
U <- ama$Uvalue

###### Calculate the null distribution. It is repeated 10*P times
frec <- matrix(0,k,1) # vector of frecuencies of individuals in each population
for (i in 1:n){
  for (pob in 1:k){
      if (pert[i]==pob){
          frec[pob]<- frec[pob]+1
      }
  }
}

# Acumulate frecuencies
frecacum <- matrix(0,k+1,1);
for (pob in 2:(k+1)){
     frecacum[pob] <- frecacum[pob-1]+frec[pob-1];
}

P <- 10*P
percentage  <- 0
for (r in 1:P){
    TW <- distrW(d, pert, np, frec, frecacum)
    partial.p.value <- 0
    for (b in 1:np){
       if (TW[b] > W0){partial.p.value <- partial.p.value +1}
    }
    partial.p.value <- partial.p.value/np
    if (partial.p.value < alpha){percentage <- percentage + 1}
}
percentage  <- percentage /P*100
```

```
out <- list(StatisticW0= W0, ProjectionsU=U,
            Percentage_under_alpha=percentage, alpha=alpha )
return(out)
}
```

```
maxW_k <- function(x_d,v_d, pert){
######################### Used in INCAnumclu ################################
#                   When number of clusters k>=2
# Input:
# x_d <- distance matrix of the partition considered as populations
# v_d <- distance from each individual of the testing cluster to the rest.
# pert <- partition of individual considered as populations.
# Output:
# contar: number of individuals of the testing cluster considered as atipical
#         or well classified.
#############################################################################


nt <- dim(v_d)[1]
k <- max(pert)
nn <- dim(x_d)[1]
atipico <- matrix(0,nt,1)




vg <- vgeo(x_d, pert);


 delta <- deltas_simple(x_d,vg,pert)


# Calculo de W para los individuos de xx/d_x

TW <- matrix(0,nn,1)+999;
for (l in 1:nn){

    phi <- proxi_simple(x_d[l,],vg,pert)

    TW[l] <- estW_simple(vg,delta, phi)$Wvalue

}


# Calculate  maxumum of  W for ind. in  xx
M <- max(TW)

###############################################
# Calculate W for ind. in  v_d and evalutate whether is atypical or not

for (ind in 1:nt){
    phi <- proxi_simple(v_d[ind,],vg,pert)
    W0 <- estW_simple(vg,delta,phi)$Wvalue

    # Clasificacion en atipico o no atipico

    if (W0<=M){
        atipico[ind] <-0
```

```
    }
    if (W0 > M){
        atipico[ind] <-1
    }

} #for ind=1:nt


contar <- 0 # To count atypical units in v_d
contar <- sum(atipico)


#out <- list(Wx=TW, Ati=atipico, suma=contar)
out<-contar

return(out)

} # end function
```

```
maxW_k1 <- function(x_d,v_d){
########################## Used in INCAnumclu ##################################
#                   When number of clusters k=1
# Input:
# x_d <- distance matrix of the partition considered as populations
# v_d <- distance from each individual of the testing cluster to the rest.
# pert <- partition of individuals considered as populations.
# Output:
# contar: number of individuals of the testing cluster considered as atipical
#         or well classified.
########################################################################
nt <- dim(v_d)[1]
k <- 1
nn <- dim(x_d)[1]
atipico <- matrix(0,nt,1)

# As there is an unique cluster in population
pert <- matrix(1,nn,1)
vg <- vgeo(x_d, pert);

# Calculate W for ind. in xx/d_x
TW <- matrix(0,nn,1)+999;
for (l in 1:nn){
     phib <- proxi_simple(x_d[l,],vg,pert);
     TW[l] <- phib              #this case W is the proximity function
}

# Calculate the maximum of W for ind. in xx
M <- max(TW)
#-----------------------------------------------------------------------
# Calculate W for ind. in v_d and evaluter whether is atypical or not
 for (ind in 1:nt){
    W0 <- proxi_simple(v_d[ind,],vg,pert)
    # Classification
    if (W0<=M){
        atipico[ind] <-0
    }
    if (W0 > M){
        atipico[ind] <-1
    }
} #for ind=1:nt

contar <- 0 # To count atypicals in  v_d
contar <- sum(atipico)
return(contar)
} # end function
```

```
path.d <- function(d){
############################ Path distance#  ###################################
# Input:
# d: distance matrix of objects in cluster C.
#
# Output:
# v_sol: path distance of C.
#############################################################################
eps <- 1E-8
d <- as.matrix(d)
n <- dim(d)[1] # number of objects

vmin <- 0
vmax <- 0

orden2 <-matrix(0,nrow=n, ncol=n)
ordent <- rep(0, n^2)
at <- rep(0, n^2)

if (n ==1){v_sol <- 0
} else {
  if (n==2){v_sol <- d[1,2]
  } else{
  for (i in 1:n){
    orden <- 1:n
    a <- d[i,]
    orden <- order(a)
    for (j in 1:n){
        orden2[i,j] <- orden[j]
        ordent[n*(i-1)+j] <- n*(i-1)+j
        at[n*(i-1)+j] <- d[i,j]
    }
    if (a[orden2[i,2]] > vmin+eps){
        vmin <- a[orden2[i,2]]
        i_min <- i
        index_min <- n*(i-1) + 1
    }
}

#Binary search
#  Adjust the extrem values for the binary search
    ordent <- order(at)
    index_min <- 1
    index_max <- n*n
    while ((at[ordent[index_min+1]] <= vmin+eps)&(index_min < index_max)){
        index_min <- index_min+1
    }
    vmin <- at[ordent[index_min]]
    vmax <- at[ordent[n*n]]
    while ((abs(at[ordent[index_max-1]] - vmax) <= eps)&
           (index_max > index_min)){index_max <- index_max -1}
    vmax <- at[ordent[index_max]]
    vs1 <- vmax
```

```
    vs2 <- vmax



    # Begin the search
    while (vmin < vmax){
        v_try <- vmin + (vmax-vmin)/2
        index_try <- index_min
        while (at[ordent[index_try+1]] <= v_try +eps){
            index_try <- index_try+1
        }
        v_try <- at[ordent[index_try]]

        romper <- 0


        aux <- haysolucion(v_try, d)
        v_try <- aux$v_try
        romper <- aux$romper


        if (romper==1){
            index_max <- index_try
            v_max <- v_try
            while ((abs(at[ordent[index_max-1]]-v_max) <= eps)&
                   (index_max > index_min)){index_max <- index_max-1}
            vmax <- at[ordent[index_max]]
            v_sol <- v_try
        } else{
            index_min <- index_try+1
            vmin <- at[ordent[index_min]]
              while (at[ordent[index_min+1]] <= vmin +eps){
                index_min <- index_min + 1}
        }
    } # while (vmin < vmax){

 } # } else{ corresponding to if (n==2){v_sol <- d[1,2]
}  # else{ corresponding to if (n ==1){v_sol <- 0


return(v_sol)
} # function
```

```
proc2 <- function(X,Y){
##### Procrustes statistic (two configurations) ##############################
# Input:
# X: =c(x1,..., xT) non standarized configuation
# Y: =c(y1,...,yT) non standarized configuation
# Output:
# M: value of the procrustes statistic (k=2)
##############################################################################

tpoints <- length(X)
X <- array(X, dim=c(tpoints, 1))
T <- length(Y)
Y <- array(Y, dim=c(T, 1))


if(T != tpoints) {stop("Configurations must have same length")}

conf <- array(0, dim=c(tpoints,2,2))

conf[,1,] <- 1:tpoints
conf[,2,1] <- X
conf <- array(conf, dim=c(tpoints, 2, 2))
conf[,2,2] <- Y

conf_s <- center(conf)

X_s <- conf_s[,,1]
Y_s <- conf_s[,,2]

# s=trace(X'*Y)

s <- sum(diag(t(X_s)%*%Y_s))

m2 <- 1-s*s


return(m2)

} #function
```

```
prock <- function(dat){
# ######################## prock ############3
#     Calculates de procrustes statistic for k>2 configurations given in dat
# Input:
# dat: each file contains measurements during the considered period of time
#
# Output:
# Mmean: procrustes weighted mean.
# scales: scaling coefficients
# Mvalue: generalized procrustes statistic
#########################################################################

dat <- as.matrix(dat)

nrep <- dim(dat)[1]   # number of configurations
tpoints <- dim(dat)[2]
p <- 2

data <- array(0, dim=c(tpoints, 2, nrep))
data[,1,] <- 1:tpoints
for (r in 1:(nrep)){
    data[,2,r] <- dat[r,1:tpoints]
}


dat_s <- center(data)

#nrep<-dim(dat_s)[3]
#tpoints<-dim(dat_s)[1]
#p<-dim(dat_s)[2]


#S=(trace(Xi'*Xj))_ij matrix

S <- matrix(0, nrep, nrep)
for (i in 1:nrep){
     for (j in 1:nrep){
         S[i,j] <- sum(diag(t(dat_s[,,i])%*%dat_s[,,j]))
     }
}

vp <- eigen(S)
mu<-vp$values

vec<-vp$vectors      #first vector are the scales we are looking for

# Normalize to have K=nrep norm
a=t(vec[,1])%*%vec[,1]
s<-abs(sqrt(nrep/a)*vec[,1])
#t(s)%*%s
```

```
##############################################3
# Value of M
# Mean configuration G
G<-matrix(0, nrow=tpoints,ncol=p)
for (i in 1:nrep){
G<- G + s[i]*dat_s[,,i]
}
G <- 1/nrep*G


#Residual sums
M <- 0
for (i in 1:nrep){
D<- s[i]*dat_s[,,i]-G
M<- M + sum(diag(t(D)%*%D))
}
M <- nrep*M


out=list( Mmean=G, scales=s, Mvalue=M)

return(out)
}
```

```
prock_array <- function(dat){
################## prock (with data arraged in a array) ###############
#     Calculates de procrustes statistic for k>2 conf. given in dat
# Input:
# dat: array structure!!!
#
# Output: generalized procrustes statistic
#############################################################################
data <- dat
at_s <- center(data)

nrep<-dim(dat_s)[3]
tpoints<-dim(dat_s)[1]
p<-dim(dat_s)[2]

#S=(trace(Xi'*Xj))_ij matrix
S <- matrix(0, nrep, nrep)
for (i in 1:nrep){
   for (j in 1:nrep){
      S[i,j] <- sum(diag(t(dat_s[,,i])%*%dat_s[,,j]))
   }
}

vp <- eigen(S)
mu<-vp$values

vec<-vp$vectors      #first vector are the scales we are looking for

# Normalize to have K=nrep norm
a=t(vec[,1])%*%vec[,1]
s<-abs(sqrt(nrep/a)*vec[,1])
#t(s)%*%s

#-------- Value of M --- Mean configuration G
G<-matrix(0, nrow=tpoints,ncol=p)
for (i in 1:nrep){
G<- G + s[i]*dat_s[,,i]
}
G <- 1/nrep*G

#Residual sums
M <- 0
for (i in 1:nrep){
D<- s[i]*dat_s[,,i]-G
M<- M + sum(diag(t(D)%*%D))
}
M <- nrep*M

out=list( Mmean=G, scales=s, Mvalue=M)
return(out)
}
```

```
proxi<- function(d, dx0, pert="onegroup"){
#################### Calculates the proximities ##############################
# Input:
# d: distance matrix between individuals (nxn)
# dx0: vector of length n with distancies from the specific individual to the
#      individuals of different groups.
# pert: integer vector indicating the group each individual belongs to.
#
# Output:
# phi: vector of proximities from the specific individual to each
#      cluster
##############################################################################
d <- as.matrix(d)
n<-dim(d)[1]
if (pert[1]=="onegroup"){pert <- rep(1,n)}
pert <- as.integer(pert)
k<-max(pert)

# We need the geometrical variabilities
var <- vgeo(d,pert)

# We need the squared distances
d <- d*d
dx0 <- dx0*dx0

phi <- matrix(0,k,1)

frec <- matrix(0,k,1) # vector of frequencies of individuals in each population

for (i in 1:n){
  for (pob in 1:k){
      if (pert[i]==pob){
          frec[pob]<- frec[pob]+1
      }
  }
}

for (i in 1:n){
  for (pob in 1:k){
     if (pert[i]==pob){
       phi[pob]=phi[pob]+dx0[i]
     }
  }
}

for (pob in 1:k){
       phi[pob]=phi[pob]/frec[pob]-var[pob]
}

return(phi)
}
```

```
proxi_simple<- function(dx0, var, pert){
#################### Proximity function (faster version)#########################
# Input:
# dx0: vector of dist. from one individual to the ind. of different clusters.
# var: geometric variabilities.
# pert: integer vector indicating the group each individual belongs to.
# Output:
# phi: vector of proximities from one individual to each cluster.

dx0 <- dx0*dx0

n <- length(pert)
k<- max(pert)


phi <- matrix(0,k,1)

frec <- matrix(0,k,1) # vector of frequencies of individuals in each population

for (i in 1:n){
  for (pob in 1:k){
      if (pert[i]==pob){
          frec[pob]<- frec[pob]+1
      }
  }
}

for (i in 1:n){
  for (pob in 1:k){
     if (pert[i]==pob){
       phi[pob]=phi[pob]+dx0[i]
     }
  }
}

for (pob in 1:k){
       phi[pob]=phi[pob]/frec[pob]-var[pob]
}


return(phi)
}
```

```
serieprock <- function(dat, nrep, g, null.p=FALSE){
##### Calculates de procrustes statistic and orders its values in decreasing way
# Input:
# dat: (ngenes*nrep)*tpoints dimension matrix. ngenes blocks with the nrep rows,
#       each row for each replication
# nrep: number of replicates
# g: names of the objects/genes (length: ngenes)
# null.p: logical value indicating if there must be included the null profile
#         or not.
#
# Output:
# M: vector of length ngenes, with the procrustes values decreasing
################################################################################


dat <- as.matrix(dat)
g <- as.matrix(g)

N <- dim(dat)[1]
tpoints <- dim(dat)[2]

n <- N/nrep

if ( ceiling(n) != floor(n)){stop("All objects must have same
                                  number of replications")}

if (null.p){
    datos <- array(0, dim=c(tpoints, 2, nrep+1, n))
    datos[,1,,] <- 1:tpoints
    for (i in 1:n){
        paso1 <- nrep*(i-1)
        paso2 <- nrep*i
        for (r in 1:(nrep)){
            datos[,2,r,i] <- dat[paso1+r,1:tpoints]
        }
        datos[,2,nrep+1,i] <- rep(1,tpoints)
    }

    M <- matrix(0, nrow=n, ncol=1)

    for (i in 1:n){
          M[i] <- prock_array(datos[,,,i])$Mvalue
    }
    #rownames(M) <- g
    o <- order(M, decreasing=TRUE)
    Mord <- M[o]
    Mord <- matrix(Mord, nrow=n)
    rownames(Mord) <- g[o]
    colnames(Mord) <- "with.nullprofile"


}else{
    datos <- array(0, dim=c(tpoints, 2, nrep, n))
```

```
    datos[,1,,] <- 1:tpoints
    for (i in 1:n){
        paso1 <- nrep*(i-1)
        paso2 <- nrep*i
        for (r in 1:nrep){
            datos[,2,r,i] <- dat[paso1+r,1:tpoints]
        }
    }
    M <- matrix(0, nrow=n, ncol=1)
    for (i in 1:n){
          M[i,] <- prock_array(datos[,,,i])$Mvalue
    }
    #rownames(M) <- g
    o <- order(M,  decreasing=TRUE)
    Mord <- M[o]
    Mord <- matrix(Mord, nrow=n)
    rownames(Mord) <- g[o]
}


out <- list(Mvalues = Mord)
return(out)
}
```

```
varb <- function(dat,  np){
###### Calculates the variance of differences abs(M(boot)-M0) #################
######       based on bootstrap sampling
# Input:
# dat: array corresponding to ONE object/gene
# np: size of bootstrap sample
# Output:
# vardif: var( abs(M(boot)-M0))
################################################################################


dif<-matrix(0, np,1)
# TM<-matrix(0, np,1)

 # Calculate procustes statistic with original replicates
M0 <- prock_array(dat)$Mvalue


nrep<-dim(dat)[3]
tpoints<-dim(dat)[1]
p<-dim(dat)[2]


for (b in 1:np){
    # Copy all replicates
    dat_b <- dat
    for (i in 1:nrep){
        for (j in 1:tpoints){
           # choose one at random
           elijo <- round(runif(1,0.5, nrep+0.5))
           dat_b[j,2,i] <-dat[j,2,elijo]
        }
    }


  # dat_b contains nrep replicates from the original ones

  # Calculate procrustes statistic and compare with M0
dif[b]<-abs(prock_array(dat_b)$Mvalue-M0)
#TM[b] <- procrustes_k(dat_bs)$Mvalue


} # for (b in 1:np)

vardif <- var(dif)

return(vardif)
} #function
```

```
varepli <- function(dat, nrep, g,  np){
#########  Measures the variability between replicates ######################
# Input:
# dat: (ngenes*nrep)*tpoints dimension matrix. ngenes blocks with the nrep rows,
#       each row for each replication.
# nrep: number of replicates.
# g: names of the objects/genes (length: ngenes)
# np: number of bootstrap replications.
#
# Output:
# nsample: bootstrap sample size
# variability: vector of length ngenes, with the procrustes values and
#              variabilities in decreasing
#############################################################################
dat <- as.matrix(dat)
g <- as.matrix(g)
N <- dim(dat)[1]
tpoints <- dim(dat)[2]
n <- N/nrep

if ( ceiling(n) != floor(n)){stop("All objects must have same
                                  number of replications")}

datos <- array(0, dim=c(tpoints, 2, nrep, n))
datos[,1,,] <- 1:tpoints
for (i in 1:n){
     paso1 <- nrep*(i-1)
     paso2 <- nrep*i
     for (r in 1:nrep){
         datos[,2,r,i] <- dat[paso1+r,1:tpoints]
     }
}

 # Calculate procrustes of original replicates
M0 <- matrix(0, nrow=n, ncol=1)
  for (i in 1:n){
      M0[i,] <- prock_array(datos[,,,i])$Mvalue
  }

nrep<-dim(datos)[3]
tpoints<-dim(datos)[1]
p<-dim(datos)[2]

varD <- matrix(0, nrow=n, ncol=1)  # vector containing the variance of the
                                   #absolute deviations
for (i in 1:n){
    varD[i] <- varb(datos[,,,i], np)
}

o <- order(varD, decreasing=TRUE)
M0ord <- M0[o]
varDord <- varD[o]
gord <- g[o]
```

```
S <- cbind(MOord, varDord)
rownames(S) <- gord
colnames(S) <- c("Mvalue", "Var_dif")

out=list(nsample=np, variability=S)

return(out)
} #function
```

```
vgeo<-function(d,pert="onegroup"){
############ Calculates the geometric variabilities ##################
# Input:
# d: distance matrix between n individuals
# pert: integer vector indicating the group each individual belongs
#
# Output: vector of geometrical variabilities of each group
#########################################################################

d <- as.matrix(d)
n <- dim(d)[1] # there are n elements
if(pert[1]== "onegroup") {pert <- rep(1,n)}
pert <- as.integer(pert)
k <- max(pert) # there are k populations

# We need squared distances
d <- d*d


var<-matrix(0,k,1) #vector of geometric variabilities

frec <- matrix(0,k,1) # vector of frecuencies of individuals in each population

for (i in 1:n){
  for (pob in 1:k){
      if (pert[i]==pob){
          frec[pob]<- frec[pob]+1
      }
  }
}

for (i in 1:n){
   for (j in 1:n){
      if(pert[i]==pert[j]){
         var[pert[i]]=var[pert[i]]+d[i,j]
      }
   }

}


for (pob in 1:k){
    var[pob] <- var[pob]/(2*frec[pob]*frec[pob])
}


return(var)

}
```