Universitat Autònoma de Barcelona

2655 BIOINFORMÁTICA:INTERFAZ WEB PARA ESTUDIAR EL EFECTO DE DIFERENTES CONDICIONES SOBRE LA EXPRESIÓN DE LOS GENES

Memoria del Proyecto Final de Carrera
de Ingeniería Informática
realizado por
José Fernández Márquez
y dirigido por
Jordi Gonzàlez Sabaté
y codirigido por
Mario Huerta
Bellaterra, 31 de Enero de 2011

Universitat Autònoma de Barcelona

etse

Escola Tècnica Superior d'Enginyeria

El sotasignat, Jordi Gonzàlez i Sabaté

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en José Fernández Márquez

I per tal que consti firma la present.

Signat:

Bellaterra, 31 de Gener de 2011

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Mario Huerta

de l'empresa, Institut de Biotecnologia i de Biomedicina de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat en l'empresa sota la seva supervisió mitjançant conveni amb la Universitat Autònoma de Barcelona.

Així mateix, l'empresa en té coneixement i dóna el vist-i-plau al contingut que es detalla en aquesta memòria.

Signat:

Bellaterra, 31 de Gener de 2011

Índice de contenido

1.- Introducción

1.1.- Motivación del proyecto

Este trabajo está englobado en un proyecto mayor que se está realizando en el IBB(Institut de Biotecnología y de Biomedicina de la UAB) para el análisis de datos de microarray . La consecución de los objetivos marcados ofrecerá al usuario una serie de herramientas vía web que permiten realizar comparaciones entre genes, estudiar sus niveles de expresión en determinadas condiciones, etc...

El desarrollo del proyecto me aportará además de formación informática, formación biotecnológica y formación matemática.

Formación biotecnológica porque me permite adentrar en el campo de la genética y experimentar cómo a partir de una matrices de datos (las microarrays) extraídas al someter una serie de genes a unas determinadas condiciones muestrales se pueden inferir estados celulares, y lo más importante, que a partir de aquí se pueden buscar genes marcadores que influyan directamente en estos estados celulares.

Formación matemática porque para obtener estas conclusiones biológicas es necesaria una base matemática, concretamente en métodos de agrupación para encontrar los posibles estados celulares y en el cálculo de intervalos de confianza para buscar los genes marcadores.

Si a la formación a adquirir le añadimos que gracias a este proyecto se puede facilitar la investigación genética(estudio de enfermedades y desarrollo de nuevos fármacos) puedo decir que la realización del mismo es enriquecedor tanto a nivel personal como formativo. Me enorgullece pensar que puedo ayudar con mi trabajo a la detección de genes responsables de enfermedades como el cáncer.

Finalmente, el hecho de poder aplicar los conocimientos teóricos adquiridos durante la carrera a un problema real hace aun más interesante la realización de este trabajo puesto que siempre se tiende a pensar que la mayoría de los conocimientos no van a ser de provecho en el mundo real y en una economía de la información como en la que vivimos, eso no es ni mucho menos cierto.

1.2.- Estado del arte

La informática del S.XXI ha avanzado de tal manera que ha permitido a ciencias como la Biología molecular avanzar de manera significativa en sus investigaciones. Por un lado la capacidad de almacenamiento de datos de hoy en día nos permite contar simultáneamente con una gran cantidad de datos. Por otra parte, el avance a nivel computacional permite realizar multitud de operaciones que ayudan al análisis y tratamiento de una gran cantidad de datos. Algo que antes no era posible.

Un ejemplo claro de estos avances es el análisis de los datos generados por tecnología de microarray. Esta tecnología consiste en exponer una gran cantidad de genes a diferentes condiciones (que son el objeto del estudio) con la intención de medir los niveles de expresión de los genes bajo esas condiciones. El resultado es una matriz numérica bidimensional donde una de las dimensiones son los genes y la otra las diferentes condiciones muestrales a las que se expusieron. Cada una de las celdas de la matriz representará entonces el nivel de expresión de un determinado gen bajo una cierta condición muestral.

Sobre los datos obtenidos se pueden realizar diferentes tipos de análisis. Uno de los más habituales consiste en agrupar las condiciones muestrales en clústers (grupos). De esta forma se podrá estudiar el efecto que tiene cada uno de los grupos en el nivel de expresión de los genes y entender así el significado biológico de las condiciones muestrales como grupo. Esta información provee un panorama general de los factores que controlan el metabolismo celular, desde la patogenicidad bacteriana hasta la susceptibilidad de los tejidos a diversas enfermedades (ej. los distintos tipos de cáncer).

El hecho de poder agrupar las condiciones muestrales permite identificar que genes tienen un nivel de expresión más representativo en cada grupo de condiciones muestrales. Estos serán los genes marcadores. Por ejemplo, si tenemos un grupo de condiciones muestrales que representan un tipo de cáncer podemos encontrar los genes que más se expresan en estas condiciones y por lo tanto saber que estos genes son probablemente los causantes de este tipo de cáncer (es decir, los genes marcadores de la patología).

Estos métodos de agrupación se pueden dividir en agrupación supervisada y no supervisada. El primer método necesita de un "profesor" que ajuste los parámetros de agrupación hasta encontrar una agrupación óptima. En cambio, la no supervisada es automática , es decir , el mismo método se encarga de ajustar los parámetros hasta llegar al óptimo. Este último método es el que se usará ya que la agrupación de las condiciones muestrales de la microarray debe realizarse de forma automática sin interacción del usuario.

Algunos de los métodos de agrupación más utilizados en el análisis de microarrays son: biclustering, Self-organizing Maps (SOM), k-means, Self-organizing Tree Algorithm (SOTA), hierarchical clustering (HC) y Partitioning Around Medoids (PAM). También se realizan métodos de reducción de dimensiones previos a la agrupación con el objetivo de reducir el nº de variables a considerar. Los métodos más utilizados para el análisis de microarrays son: las componentes principales (PC) y el multi-dimensional scaling (MDS). Estos son los métodos que se usarán en el aplicativo desarrollado tras recibir asesoramiento de Mario Huerta y Juan Antonio Cedano, ambos con amplia experiencia en el análisis de microarrays.

Para conocer el grado de fiabilidad de una agrupación resultante existen los cálculos de integridad. Hay muchos índices que miden la fiabilidad de una agrupación, en nuestro aplicativo usaré el índice Dunn y Silhouette Width, aunque también sometemos a estudio otros índices como Hartigan, Calinsky y la conectividad de los clústers. Estos índices se escogen bajo supervisión de Mario Huerta y Juan Antonio Cedano y tras comprobar que son los que daban mejores resultados.

Existen varias librerías estadísticas que facilitan el uso de los métodos de agrupación comentados. Los más usados son CLUTO[1]y determinados paquetes de R-STATISTICS[2].

R-STATISTICS nos ofrece diferentes paquetes para el análisis de microarrays. Estos paquetes se encuentran en el repositorio Comprehensive R Archive Network (CRAN[3]) y algunos de los que analizaremos son:

- pcaPP (PC)[4]: escalado PC de una matriz de datos
- som (SOM) [5]: agrupación SOM sobre una matriz de datos
- smacof (MDS)[6] : escalado MDS de una matriz de datos
- cdmdscale[7] : herramienta que se incluye en la librería estándar de R-STATISTICS que realiza el escalado MDS de una matriz de datos.
- biclust (*biclustering*)[8]: *biclustering* de una matriz de datos
- clusterSim[9]: cálculo de varios índices de integridad para una agrupación determinada
- isa2[10]: biclustering ISA de una matriz de datos
- clValid[11]: herramienta que realiza varios tipos de agrupación de una matriz de datos así como el cálculo de varios índices de integridad.
- impute[12] : rellenar los posibles huecos que haya en la matriz de datos.
- gtools[13] : ofrece métodos de estadística tales como la combinatoria.

Estos métodos de agrupación se incorporarán en la aplicación web (applet[i]) desarrollada por el IBB para el análisis de microarrays(PCOPGene[14][39]) . De esta se facilitará el análisis conjunto de las dependencias de expresión, en las que se centran dichas herramientas, con los métodos de agrupación de condiciones muestrales clásicos.

Esta aplicación web está alojada en el servidor para el análisis de microarrays: http://revolutionresearch.uab.es[38]

El PCOPGene es una herramienta que nos permite estudiar la relación de expresión entre genes bajo distintas condiciones muestrales, clasificar estas condiciones y estudiar sus efectos en diferentes relaciones[36] . Esta herramienta, a partir de las microarrys de entrada calcula las *Principal Curves of Oriented Points* (PCOP[15]), después de analizar estas PCOP podemos determinar el tipo de relación a nivel de expresión entre los genes de la microarray y gracias a su potente interfaz web estudiar las fluctuaciones

i) *Un **applet** es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin*

de esas dependencias de expresión en detalle[37]. Ahora, con el nuevo aplicativo a desarrollar, podrán consultarse además el efecto de las condiciones muestrales agrupadas en clústers sobre dichas fluctuaciones.

1.3.- Objetivos

El objetivo principal es añadir algunos de los algoritmos de agrupación más utilizados para el análisis de microarrays a una aplicación web de análisis de microarrays ya existente. De esta manera los usuarios dispondrán de un paquete de herramientas que les permitirán estudiar el efecto de los diferentes clústers de condiciones muestrales sobre la expresión de los genes y entender así el significado biológico de los clústers obtenidos.

Cada algoritmo de agrupación generará su propio fichero de clústers por lo tanto en el desarrollo de la aplicación también se desarrollará un gestor de ficheros clústers a través de la web para facilitar su uso.

A parte, se agregará una herramienta de búsqueda de genes en función de la distribución de los clústers a lo largo de la expresión del gen. Esta búsqueda se basará en el cálculo para cada gen de las distribuciones normales de los clústers y sus intervalos de confianza derivados de la distribución T de Students[i].

1.4.- Organización de la memoria

Para alcanzar los objetivos propuestos se ha seguido la planificación que se expone a continuación.

El trabajo realizado tuvo una duración de un año lectivo,se finalizó durante el año académico 2009/2010 y se divide en dos bloques diferenciados.

El primero y principal es el que se encarga de realizar las diferentes agrupaciones de condiciones muestrales para una microarray dada. Estas agrupaciones agrupan las condiciones muestrales en grupos de condiciones muestrales con similares características.

Puesto que lo que interesa es poder agrupar condiciones muestrales por similitud para después encontrar los genes marcadores (genes más representativos del clúster o grupo) se tiene que encontrar un paquete estadístico lo más versátil y completo posible. Versátil en el sentido de que mantenga compatibilidad con algunos de los lenguajes de programación más comunes y completo en el sentido de que puedan calcularse todos los métodos de agrupación que se pretenden implementar en la aplicación.

Obtenidas las diferentes distribuciones de clústers para una microarray se pretende encontrar los que tengan un grado de fiabilidad mayor. Para ello se han usado los valores de integridad. Se realizaron una serie de pruebas para escoger los índices de integridad más adecuados.

Estos nuevos módulos se han integrado en un preproceso ya existente que se ejecuta al cargar las microarrays en el sistema.

Seguidamente, se ha desarrollado una herramienta web para la gestión de estos resultados y se ha integrado en la aplicación web existente : PCOPGene[14], para que estas distribuciones de clústers sean accesibles y seleccionables por los usuarios y una vez seleccionadas utilizadas por el resto de herramientas del servidor[38].

El segundo bloque consistía en desarrollar una herramienta de búsqueda para los genes de la microarray que se expresen de la forma requerida por el usuario para la distribución de clústers seleccionada. Con esta herramienta el usuario podrá encontrar los genes marcadores para una agrupación de condiciones muestrales determinada .

Realizada la implementación de este bloque se integra la nueva funcionalidad en la web PCOPGene ya existente .

Cada bloque o fase ha estado sujeta a diferentes modificaciones durante su implementación para mejorar su funcionalidad y reusabilidad, y operativilidad, usabilidad y entendibilidad en el caso de las interficies web, con la ayuda y asesoramiento del codirector Mario Huerta.

i) *T de Students es una distribución de probabilidad que surge del problema de estimar la media de una población normalmente distribuida cuando el tamaño de la muestra es pequeño.*

Los capítulos que siguen a continuación describen en detalle el proyecto y trabajo realizado. Estos son:

-Fundamentos teóricos: se definen los fundamentos teóricos que son la base del análisis realizado. Fundamentos tanto a nivel biológico como a nivel computacional.

-Fases: Las diferentes fases de la estrategia seguida para alcanzar los objetivos preestablecidos para cada fase (agrupación y búsqueda de genes). Se exponen los problemas encontrados durante el desarrollo así como las soluciones y decisiones tomadas.

-Informe técnico: se describen las implementaciones que han sido necesarias en cada una de las fases, esto incluye algoritmos y diagramas de flujo.

-Conclusiones

-Bibliografía

-Resumen

2.- Fundamentos teóricos

El análisis realizado se fundamenta básicamente en dos aspectos, el biológico y el computacional.

El biológico hace referencia a los genes, a su expresión y como estos llevan a los diferentes estados celulares.

En el aspecto computacional se definen conceptos relacionados con la tecnología de microarrays, agrupación, cálculos de integridad e intervalos de confianza en los cuales se basa la búsqueda de genes.

2.1.- Introducción biológica

La biología es una disciplina científica que abarca un amplio espectro de campos de estudio que, a menudo, se tratan como disciplinas independientes . Este trabajo se centra en la biología molecular, más concretamente el la genética molecular

La genética molecular estudia el ADN[i], su composición y la manera en que se duplica. Asimismo, estudia la función de los genes desde el punto de vista molecular .

Al lenguaje universal de la vida se le conoce como código genético ya que expresa cómo funcionan , cómo son y cómo han de desarrollarse todos los organismos, desde los más simples a los más complejos. Para distinguir entre organismos existe el genoma , que no es más que un *manual de instrucciones* propio de cada organismo.

Los genomas de cada individuo dentro de una misma especie son ligeramente diferentes. Este conjunto de instrucciones se almacena en cada una de las células del organismo como si de una biblioteca se tratase donde los libros serían los genes y el ADN el lenguaje usado. Estos genes contienen toda la información que el individuo necesita para formarse, nacer, vivir y reproducirse. En determinadas ocasiones los genes se almacenan en estanterías formando los cromosomas.

Los dos cometidos principales que desarrolla la información que contiene el genoma son: crear proteínas encargadas tanto del desarrollo como del funcionamiento de cualquier organismo y la herencia, es decir, almacenar la información para después transmitirla a la descendencia.

Podemos decir que el genoma encierra el secreto de la vida. Pero eso es solo el comienzo, ya que el futuro de cada organismo dependerá de más factores externos como factores físicos, biológicos e incluso culturales en el caso de los seres humanos.

Denominamos expresión génica al proceso de transformación de la información codificada en los ácidos nucleicos[ii] en las proteínas que intervienen en el desarrollo y funcionamiento de todo organismo. No todos los genes se expresan al mismo tiempo ni en todas las células. Sólo los genes llamados

i) *El ácido desoxirribonucleico, frecuentemente abreviado como ADN (y también DNA, del inglés DeoxyriboNucleic Acid), es un tipo de ácido nucleico, una macro molécula que forma parte de todas las células*

ii) *ADN y ARN (ácido ribonucleico)*

housekeeping (marcadores) se expresan en todas las células del organismo y codifican proteínas esenciales para el funcionamiento general de las células. El resto de genes se expresan o no en los diferentes tipos de células, dependiendo de la función de la célula en un tejido particular. Por ejemplo, genes que codifican proteínas responsables del transporte axonal se expresan en neuronas pero no en linfocitos en donde se expresan genes responsables de la respuesta inmune. También existe especificidad temporal, esto quiere decir que los diferentes genes en una célula se encienden o se apagan en diferentes momentos de la vida de un organismo o ciclos celulares. Además, la regulación de los genes varía según las funciones de éstos.

La expresión de un gen se obtiene cuando una porción de ADN se transcribe para crear una molécula de ARN como primer paso en la síntesis de proteínas. Del numeroso grupo de genes que se encuentran en el núcleo y que codifican proteínas, sólo un subconjunto de ellos se expresará en un momento determinado. Esta expresión selectiva viene regulada por distintos aspectos: tipo de célula, fase de desarrollo del ser vivo, estímulo interno o externo, enfermedades, etc.

El estudio de la expresión genética puede ayudar a responder una gran cantidad de cuestiones biológicas. Por ejemplo , encontrar los genes que se expresan en un cierto ciclo celular o en la producción de proteínas. Otra de las cuestiones importantes serían las investigaciones sobre el efecto de enfermedades en la expresión genética.

Para responder a estas cuestiones se han usado tecnologías capaces de obtener, almacenar y analizar toda la información. El problema de estas tecnologías es que se basan normalmente en el análisis de un solo gen sometido a varias condiciones muestrales . En los últimos años han aparecido unas tecnologías capaces de obtener datos simultáneamente de una gran cantidad de genes. Las ventajas de tener esta gran cantidad de datos respecto el análisis individual de un gen son :

·La capacidad de identificar los genes que se expresan de manera desmesurada en un cierto ciclo celular

·Encontrar patrones similares de comportamiento en determinadas condiciones muestrales o encontrar un conjunto de genes que reaccionan de forma inversa ante determinados estímulos (condiciones muestrales) .

La tecnología más usada para este tipo de análisis es la creación de *Microarrays.*

2.2.- Bioinformática

Una de las definiciones más sencillas de la bioinformática corresponde a la aplicación de la tecnología de computadores a la gestión y análisis de datos biológicos. La bioinformática hace referencia a varios campos de estudios que necesitan el desarrollo de técnicas que incluyen, informática, matemática aplicada,estadística, bioquímica, química, ciencias de la computación e inteligencia artificial para solventar problemas, analizar datos o simular sistemas, todos de tipo biológico.

De entre todos los objetivos que se intentan estudiar utilizando la bioinformática destacamos el estudio de la regulación genética para interpretar perfiles de expresión génica utilizando datos de una microarray.

Dado que las microarrays contienen una gran cantidad de datos se hacen necesarias técnicas que permitan estudiar estos datos de un modo más global. La técnica mas usada es la de agrupamiento o agrupación que permite agrupar genes o condiciones muestrales en grupos con similares características.

2.2.1.- Microarrays

Un chip de ADN (*del inglés DNA microarrays*) es una superficie sólida a la cual se unen una serie de fragmentos de ADN. Las superficies empleadas para fijar el ADN son muy variables y pueden ser vidrio, plástico e incluso chips de silicio. Los arreglos de ADN son utilizadas para averiguar la expresión de genes, monitorizándose los niveles de miles de ellos de forma simultanea[16].

La técnica consiste en extraer el RNAm[i] de una célula buena y de otra experimental mediante *isolation RNA*[ii], una vez extraído los dos RNAm se marca cada uno de ellos con un color distinto y se

i) *RNA mensajero, es el portador de la información genética del ADN que se utiliza para la síntesis de proteínas*
ii) *Técnica que consiste en extraer de forma pura el RNA de una sustancia biológica*

combinan los dos. Acto seguido se vierte el combinado en la superficie del chip de tal modo que cada RNAm se unirá o no a los cDNA[i] de cada gen del chip. Finalmente, aplicando técnicas de análisis de imágenes es posible generar una matriz de datos numéricos, a partir de los patrones de intensidades de cada celda y discriminando la señal informativa de ruido que pudiera haber en segundo plano. Estos datos numéricos corresponderán al nivel de expresión de cada gen expuesto a una serie de condiciones muestrales. Por ejemplo, si el ARN de la célula experimental se coloreó de color rojo, los genes con un color más cercano al rojo tienen un nivel de expresión más elevado en las células experimentales. En la *fig. 2.1* se puede ver gráficamente el proceso de creación de la microarray y en la *fig.2.2* se observa una microarray real donde cada celda tiene una intensidad de color distinta que después se usará para obtener el nivel de expresión de cada gen.

Actualmente existen diferentes bases de datos a nuestro alcance a través de internet que unifican y facilitan toda esta información genética además de ofrecer diversas herramientas para el análisis de esta gran cantidad de información. Algunas de estas bases de datos por ejemplo son las que hay en el EMBL (European Molecular Biology Laboratory), el SIB (Swiss Institute of Bioinformatics), el EBI (European Bioinformatics Institute) o el NCBI (National Center for Biotechnology Information). El EBI[17] y el NCBI[18] son los que más información contienen y por lo tantos los más utilizados.

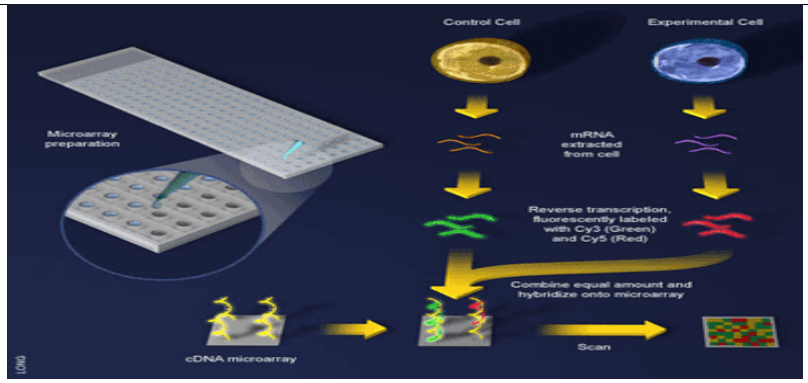


Figura 2.1: Creación de la microarray. Primero se extrae el RNAm de dos células, una normal y otra experimental. Después se identifica a cada una de ellas mediante fluorescencia, se combinan y se vierte encima de la superficie del chip para que se combinen con el cDNA de los genes. Finalmente se escanea para ver los niveles de expresión según el color que adquiera cada celda (cada gen)

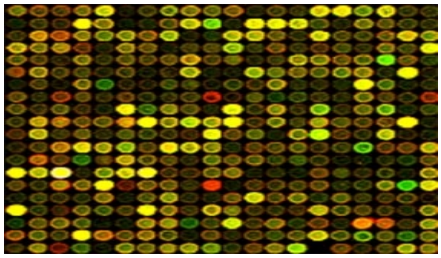


Figura 2.2: Ejemplo de microarray, cada celda tiene una intensidad de color que después se usará para obtener el nivel de expresión de cada celda,gen

Por lo tanto las microarrays son una potente fuente de obtención de perfiles de expresión de genes sometidos a diferentes condiciones, identificar los patrones de los niveles de expresión será muy útil para compararlos y poder estudiar las respuestas de los genes.

Aplicando una serie de procesos experimentales y computacionales sobre la microarray se obtiene una matriz numérica bidimensional que consta de los genes de poblaciones distintas como individuos y de las condiciones muestrales a las que se expusieron las células como variables en el caso que se quiera estudiar a los genes, o a la inversa, si es que se quiere realizar un estudio comparativo de las condiciones a que se someten[19](este estudio es en el que se basa el trabajo realizado). Cada uno de los valores de la

i) *ADN complementario, se sintetiza a partir del RNA mensajero (RNAm maduro)*

matriz representa el nivel de expresión de un determinado gen bajo una cierta condición muestral. Es posible que en algunos casos se produzcan errores en el proceso y se generen huecos, estos huecos tiene valor 0 dentro de la matriz.

Estas matrices son de grandes dimensiones puesto que existen una gran cantidad de genes y de condiciones muestrales.

Dado que realizar una análisis de esta matriz de gran dimensionalidad es una tarea prácticamente imposible se hacen necesarias técnicas computacionales que permitan agrupar todos estos datos y a partir de el agrupamiento realizar el análisis biológico. La técnica mas utilizada es el agrupamiento o *agrupación.*

2.2.2.- Agrupación

El objetivo de los algoritmos de agrupamiento (clustering en inglés) es , dada una matriz de individuos y variables, encontrar un grupo (clúster) de un conjunto de individuos, de tal forma que los clústers resultantes sean homogéneos y/o estén bien separados. El punto clave es reducir la gran cantidad de datos caracterizándolos en grupos más pequeños de individuos similares. Esto implica que los individuos pertenecientes a un mismo clúster son lo más similares posibles entre ellos, mientra que los individuos de clústers distintos son lo más disimilares posibles[19]. La agrupación de los individuos se realizará de acuerdo a la separación entre ellos determinada por una medida de distancia dada, llamada medida de disimilaridad.

La agrupación la podemos aplicar de tres formas distintas, una es realizar la agrupación directamente sobre todas las variables o columnas. Otra sería reducir previamente las variables o columnas agrupándolas por similitud de comportamiento mediante técnicas como el Multidimensional Scaling (MDS) y el Principal Components (PC) y después aplicando la agrupación. Y finalmente también se puede agrupar aplicando la agrupación simultáneamente tanto a individuos como a variables, a esta técnica se le conoce como *biclustering*.

El agrupamiento se puede dividir en dos grandes categorías: jerárquicos y de particionamiento.

· **Multidimensional Scaling (MDS)**

El objetivo principal de esta técnica es reducir la dimensionalidad de un conjunto grande de datos de modo que se facilite el estudio de éstos. En el caso de las microarrays reducir las dimensiones (variables) supone una reducción importante de tiempo en el cálculo posterior de los algoritmos de agrupación.

El MDS trata de encontrar la dimensión óptima de modo que se pierda la menor información posible. Cuanto menor sea esta dimensión más fácil será el análisis posterior de los datos. Es decir, se pretende resumir toda la información en varios capítulos (dimensiones) y cuanto menos capítulos se necesiten para resumir toda la información mejor capacidad de computo se tendrá sobre estos datos.

La idea consiste en encontrar una representación gráfica del conjunto de datos en las menos dimensiones posibles perdiendo la menor cantidad de información. Por ejemplo, si tenemos una microarray de 3 variables (genes) y X individuos (condiciones) podemos representar estos punto en un plano de 3 dimensiones (*fig. 2.3*). Si reducimos una de las dimensiones buscando 2 planos que engloben a cada una de las condiciones muestrales (*fig. 2.4*) obtendremos una representación gráfica óptima para el análisis puesto que es más fácil trabajar en 2 dimensiones que no con 3 (*fig. 2.5*).

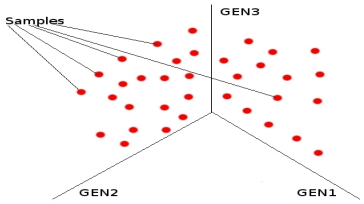


Figura 2.3: Representación gráfica de una microarray de 3 dimensiones (3 genes) y N condiciones muestrales

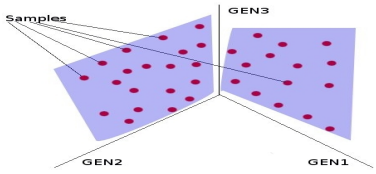


Figura 2.4: Con 2 planos podemos reducir una de las dimensiones. Estos dos planos engloban a la mayoría de condiciones muestrales

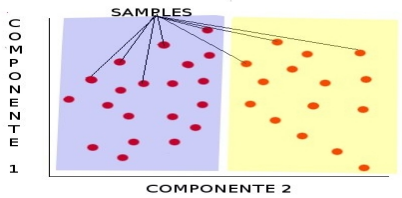


Figura 2.5: A cada plano encontrado se le conoce como componente. Representando los datos en 2 dimensiones se facilita la capacidad de análisis

Para medir la bondad del escalado se utiliza una medida llamada *stresss* . Por lo tanto, se buscan representaciones geométricas en X dimensiones de modo que el *stress* sea mínimo. Empíricamente, se considera que si el *stress* es alrededor de 0,2 la bondad del ajuste es pobre; si es del 0,05 la bondad del ajuste es buena y a partir de 0,025 es excelente.

Mediante un proceso iterativo se trata de minimizar el nivel de *stress* para un número fijo de X dimensiones con el fin de encontrar las dimensiones óptimas.

· Principal Components Analysis (PCA)

Principal components analysis es una técnica estadística utilizada para la determinación de variables claves dentro de un conjunto multidimensional de datos que explican las diferencias en las observaciones y que puede ser usada para simplificar el análisis y visualización de conjunto de datos multidimensionales.

Se trata de encontrar la cantidad mínima de variables necesarias para mostrar el máximo de información posible. Dadas *m* observaciones con *n* variables la finalidad del PCA es reducir la dimensionalidad de la matriz de datos encontrando las *r* nuevas variables que contengan el máximo posible de la varianza que había con las *n* variables originales. Cada componente principal es una combinación lineal de las variables originales y por lo tanto es posible obtener el significado de lo que la componente representa.

En el tratamiento de microarrays, si se consideran los genes como variables y las condiciones muestrales como individuos; el PCA crea un conjunto de “componentes principales de genes” que describen las características de los genes que mejor explican la respuesta experimental que ellos producen. Por ejemplo, si tenemos un conjunto de genes a los cuales se les somete a un par de condiciones experimentales disponemos el conjunto de genes en el espacio usando como ejes las dos condiciones muestrales. A partir de esta disposición se busca una primera componente principal que “resuma” mejor una de las condiciones muestrales y usando esta componente buscamos la segunda componente que mejor resuma la otra condición muestral siendo esta componente ortogonal de la primera puesto que la combinación lineal de las dos debe contener la información de *ambos(fig. 2.6, fig. 2.7).*

En resumen,a partir de la matriz de covarianza extraída de la microarray se resume la información compactándola usando las componentes principales. Estas componentes clarifican la relación entre los clústers calculados y es un punto de inicio para examinar en detalle las relaciones y diferencias entre genes.

Como que cada componente encontrada explica parte de la información que no explican las otras componentes podemos decir que la suma de todas las componentes contiene toda la información.

Sirve para hallar las causas de la variabilidad de un conjunto de datos y ordenarlas por importancia. El conjunto de todas las componentes contienen toda la información. La idea es encontrar las 2 componentes principales que contengan más información.

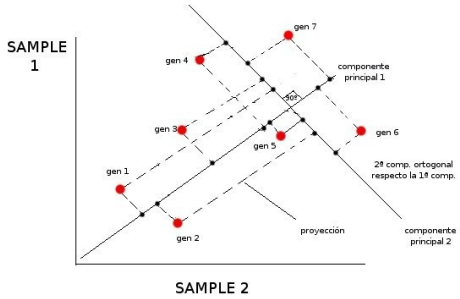


Figura 2.6: Disposición de los genes en un plano con las condiciones muestrales como ejes. Primero se busca la primera componente y a partir de ésta buscamos la segunda que es ortogonal a la primera puesto que debe ser una combinación lineal con tal de preservar toda la información

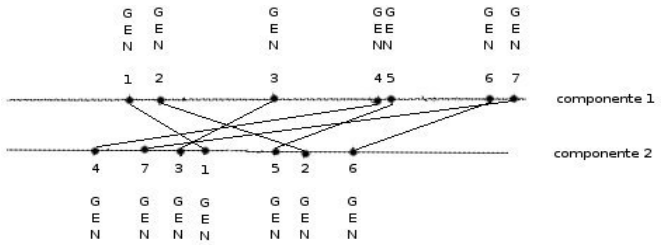


Figura 2.7: La combinación de las dos componentes contiene la totalidad de la información de todos los genes respecto las dos condiciones muestrales

· Biclustering

[20]Las principales características comunes de las técnicas de agrupación se resumen en la búsqueda de conjuntos disjuntos de genes, de tal manera que aquellos genes que se encuentren en un mismo clúster presenten un comportamiento similar frente a todas las condiciones del microarray. Además, cada uno de los genes debe pertenecer a un único clúster (y no a ninguno) al final del proceso. Las técnicas de *biclustering* se presentan como una alternativa más flexible, ya que permiten que las agrupaciones se formen no solo en base a una dimensión, sino que sea posible formar biclusters que contengan genes que presenten un comportamiento similar frente a un subconjunto de condiciones del microarray. Esta característica es muy importante, ya que aumenta la capacidad de extracción de información a partir de un mismo microarray, pudiendo ignorar determinadas condiciones frente a las cuales un grupo de genes no se co-expresen. De la misma forma, también es posible hacer *biclustering* en el sentido inverso, esto es, seleccionando características en función de un subconjunto concreto de genes.

Otro aspecto importante que diferencia a las técnicas de *biclustering* frente a las de agrupación es la forma en que las agrupaciones son hechas, ya que ahora se permite el solapamiento (varios genes pueden estar contenidos en varios biclusters a la vez), así como que existan genes (o condiciones) que no se hayan incluido en ningún subconjunto. Esta característica aporta más flexibilidad a este tipo de técnicas, ya que no obliga a incluir cada gen en una agrupación determinada, sino que un determinado gen no pertenecerá a ningún bicluster si sus valores de expresión no se ajustan a ninguno de los patrones. Asimismo, es posible que un mismo gen pertenezca a varios biclusters, si en cada uno de ellos se consideran un subconjunto de todas las condiciones muestrales, de forma que un mismo gen pueda estar participando en varias funciones celulares de manera simultánea.

En general, el problema de localizar biclusters en un microarray presenta mayor complejidad que la agrupación, ya que se plantean muchas más posibilidades a la hora de agrupar los datos.

Los distintos biclusters obtenidos pueden clasificarse según diferentes tipos (ver *fig. 2.8*), que variarán dependiendo del método concreto que se esté utilizando para su obtención. Los cuatro tipos principales se recogen a continuación:

-Biclusters con valores constantes. Este tipo de biclusters se corresponden con submatrices que contienen valores idénticos en todas las posiciones. En ellos, todos los genes presenta el mismo valor de

expresión frente a todas las condiciones contempladas.

- Biclusters con valores constantes en filas o columnas. Son biclusters cuyos genes presentan un comportamiento similar frente a las condiciones, aunque con diferentes valores de expresión para cada gen, en el primer caso. En el caso de biclusters con valores constantes en columnas, se recogen un conjunto de condiciones, donde en cada una de ellas los genes presentan el mismo valor de expresión, pero variando de una condición a otra. En este segundo caso, los genes presentan un comportamiento idéntico entre ellos.

-Biclusters con valores coherentes. Este tipo de biclusters recoge relaciones entre genes y condiciones que no tienen por qué ser directas, sino que se obtengan a partir de un análisis numérico de los datos contenidos en la matriz.

-Biclusters con evoluciones coherentes. Los biclusters que presentan evoluciones coherentes presentan una principal diferencia con respecto a los anteriores, ya que se ignoran los valores numéricos concretos para trabajar con las evoluciones o comportamiento de los datos, vistos como símbolos.

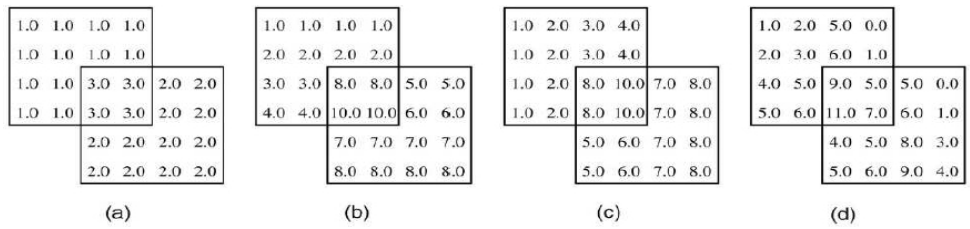


Figura 2.8: Distintos tipos de biclusters con solapamiento. (a)Valores constantes, (b) Valores constantes en filas, (c) Valores constantes en columnas, (d) Valores coherentes

· Agrupamiento jerárquico

En inglés hierarchical clustering (HC). Se generan sucesiones ordenadas (jerarquías) de clústers. Se pueden dividir clústers grandes en clústers más pequeños o se pueden juntar clústers pequeños en clústers más grandes. Su estructura se representa en forma de árbol y se la llama dendrograma. Los algoritmos jerárquicos se pueden dividir en aglomerativos y divisivos. Los aglomerativos parten inicialmente de que cada individuo es un clúster, en cambio los divisivos parten de un solo clúster al cual pertenecen todos los individuos.

- *Hierarchical clustering (HC)*

[20]Este tipo de técnicas han sido aplicadas tanto a datos de expresión génica como a otro tipo de datos en diversos contextos[21].

La mayoría siguen una estrategia aglomerativa o *bottom-up*, donde el proceso comienza con tantos clústers como genes hayan en el microarray que se analiza.

Dichos clústers se van uniendo iterativamente, con el objetivo de formar agrupaciones más grandes que contengan la mayor cantidad posible de genes similares, hasta que se cumpla una determinada condición de terminación, o todos los genes se unan en un sólo clúster.

La secuencia de los clústers que se van generando se representan mediante un árbol binario, denominado *dendrograma (fig. 2.9)*, de manera que es posible obtener el número determinado de clústers que se quiera, dependiendo del nivel del árbol al que se trabaje.

Durante el proceso de agrupación, el número de clústers siempre se va disminuyendo, proporcionando también un gran número de clústers (tantos como se quieran según el nivel del árbol).

Los genes se van agrupando en clústers de forma determinista, no pudiendo pertenecer a dos grupos distintos en el mismo nivel jerárquico.

No obstante, este tipo de métodos presentan algunos inconvenientes.

Por ejemplo, la mejor unión de dos clústers diferentes en cada etapa puede conducir a posteriores agrupaciones que no sean las óptimas. De la misma forma, las asignaciones realizadas en las distintas iteraciones no podrán ser deshechas en fases posteriores.

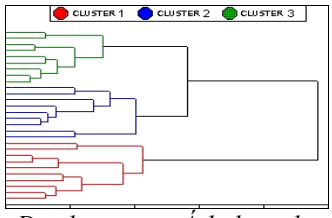


Figura 2.9: Dendrograma. Árbol resultante de un HC sobre un conjunto de datos. En este caso se puede distinguir claramente 3 grupos o clústers.

· Agrupación por particionamiento

Previamente se divide el conjunto de individuos en un numero prefijado de clústers (K) y después se van reasignado los individuos a los clústers hasta que se cumpla algún criterio de parada. Algunos algoritmos de este tipo son: K-means, Partitioning Around Medoids (PAM), Self-organizing Map (SOM) y Self-organizing Tree Algorithm (SOTA).

- *K-means*

K-means (MacQueen, 1967[22]) es un algoritmo no supervisado para agrupar un conjunto de datos a partir de un cierto número de clústers (K) fijado a priori. La idea principal es definir K centroides[i] iniciales(*fig. 2.10*), uno por cada clúster. Estos centroides iniciales pueden escogerse de varios modos:

– Al azar

– Usando las primeras K observaciones como centroides

– Escogiendo los centroides lo más separados posibles

– Escogiendo la primera observación como centroide , para escoger el segundo centroide buscar la observación que esté a una distancia dada respecto este centroide, para el tercer centroide buscar la observación que esté a la distancia dada y así sucesivamente hasta encontrar los K centroides iniciales.

La mejor opción es escogerlos lo mas separados unos de los otros.

El siguiente paso es seleccionar cada punto procedentes de los datos de entrada y asociarlos al centroide más cercano(*fig. 2.11*). Cuando hayamos recorrido todos los puntos tendremos una primera aproximación a la agrupación. En este punto recalculamos los nuevos K centroides a partir de los clústers encontrados*(fig. 2.12)*. Después de obtener los nuevos centroides podemos realizar una nueva asociación entre los puntos y los nuevos K centroides*(fig. 2.13)*. A partir de aquí volvemos a realizar los dos pasos anteriores hasta que los K centroides no cambien o el error cuadrático*(fig. 2.14)* tenga una variación mínima . En ese caso tendremos los K clústers resultantes.

Figura 2.10: Se escogen los k centroides (k= 3 en este caso) de forma aleatoria

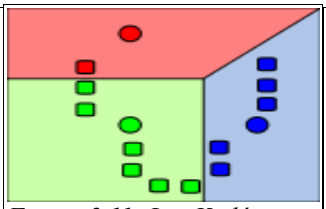


Figura 2.11: Los K clústers se generan a partir de la asociación de cada observación con el centroide más cercano.

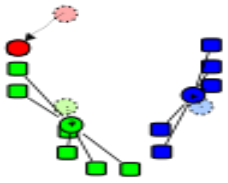


Figura 2.12: Los centroides de cada clúster son ahora las nuevas medias

i) *Un centroide corresponde a la media de todos los puntos de un clúster.*

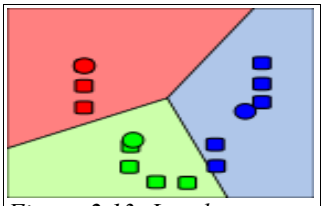


Figura 2.13: Los dos pasos anteriores se repiten hasta que no varíen los centroides

$$E = \sum_{i=1}^{K} \sum_{p \in c_i} |p - m_i|^2$$

Figura 2.14: Error cuadrático donde "p" es un punto del clúster "i", "K" es el número de clústers y "m"es la media de todos los puntos del clúster "i"

- *Partitioning Around Medoids (PAM)*

[23]Kaufman&Rousseeuw (1990) propuso el algoritmo de agrupación Partitioning Around Medoids (PAM) el cual mapeaba una matriz de distancias en un número K de clústers definidos y se basaba el algoritmo de agrupación k-medoids.

Este algoritmo k-medoids está relacionado directamente con el algoritmo k-means. Ambos son algoritmos de particionamiento y ambos tratan de minimizar el error cuadrático, la distancia entre los puntos pertenecientes al clúster y el punto designado como centro de dicho clúster. A diferencia del k-means, k-medoids escoge puntos propios del conjunto de datos como centro (medoids[i]).

El algoritmo k-medoids es más robusto a los ruidos y a valores atípicos (outlayers) en comparación con el k-means.

El primer paso consiste en escoger al azar *K* de los *n* puntos del conjunto de datos que harán de medoids.

En el segundo paso asociamos cada uno de los puntos a su medoid más cercano usando cualquier métrica válida de distancias (por ejemplo la distancia Euclidean o la distancia Manhattan).

Una vez asociados los puntos a los K clústers calculamos el coste total del sistema. Este coste total es la suma de las distancias de cada uno de los medoids respecto a sus puntos asociados (*fig. 2.15*).

En este punto realizamos el siguiente proceso:

3- Para cada *m* medoid

-Para cada punto *o* no-medoid

-Intercambiamos el medoid *m* por el punto *o* y calculamos el coste total de la nueva configuración.

4- Nos quedamos con la configuración con coste menor.

5- Repetimos el proceso a partir del segundo paso hasta que los medoids no cambien.

$$cost(x, c) = \sum_{i=1}^{d} |x - c|$$

Figura 2.15: Coste entre dos puntos. Donde x es cualquier punto del conjunto de datos, c es un medoid y d es la dimensión del punto

- *Self-organizing Map (SOM)*

En 1982 T. Kohonen presentó un modelo de red denominado mapas auto-organizados o SOM[24], basado en ciertas evidencias descubiertas a nivel cerebral.

i)*Medoids , objetos representativos de un conjunto de datos o un clúster con un conjunto de datos cuyo promedio de disimilitud de todos los objetos en el grupo es mínima .*

[20]SOM se basa en el uso de una red neuronal con un determinado número de nodos o neuronas.

Normalmente, la configuración de dichas neuronas es rectangular o hexagonal, donde los nodos poseen un valor inicial aleatorio y tienen asociado un vector del mismo tamaño que los datos de entrada, que se van ajustando durante el proceso. Una vez que la red sea estable, dichos vectores se utilizan para ir agrupando lo genes según su cercanía a los vectores de referencia. El algoritmo consiste en un proceso iterativo que se basa en la búsqueda del vector de referencia más cercano a un cierto gen escogido de modo aleatorio, al que se le denomina vector ganador, y que permitirá la actualización de los vectores de referencia, mediante el uso de una función de aprendizaje (gaussiana o de vecindad).

Debido a su componente aleatorio, SOM es un método no determinista[i], afectando de esta manera el orden en que los genes son seleccionados al resultado del algoritmo. La principal ventaja de este método es que presenta menor sensibilidad frente al ruido. Sin embargo, a diferencia de los métodos basados en agrupamiento jerárquico, en este caso es necesario especificar el número de clústers que se quieran obtener como resultado. Por lo tanto se tendrá que realizar el algoritmo para distintos números de clústers y analizar cual es el resultado óptimo mediante los índices de integridad.

- Self-organizing Tree Algorithm (SOTA)

[20]La diferencia de los algoritmos de agrupación jerárquico, que siguen una estrategia aglomerativa (o de bottom-up), es que el método SOTA sigue un esquema divisor o de top-down, donde el proceso de agrupación comienza con un árbol binario formado por un nodo principal y dos hojas, cada una de las cuales correspondientes a un clúster diferente. A partir de ahí, se extiende el árbol mediante una de las hojas, a las que se le añaden dos nuevas hojas. La elección de las hojas a expandir en cada iteración se realiza mediante un valor definido a partir de las distancias entre cada clúster y los genes contenidos en él.

Este método combina la estructura de árbol de la agrupación jerárquica con la de red neuronal utilizada en SOM.

Al igual que SOM, SOTA es un algoritmo no determinista, que recoge las ventajas del primero, añadiendo también las de la agrupación jerárquica. En este caso, el número de clústers va aumentando iterativamente, pudiendo detener el algoritmo en cualquier punto para conseguir el número deseado de agrupaciones.

En un estudio realizado en 2006[25], se presenta una comparación experimental entre HC[ii], SOM y SOTA, sobre datos de expresión genómica. Como resultado de dicho estudio se deduce que la última estrategia presentada, SOTA, combina las ventajas de los métodos de agrupación jerárquico y SOM, ya que proporciona una representación visual en forma de árbol de los clústers generados, presentando además una baja sensibilidad al ruido.

2.2.3.- Índices Integridad

Un problema fundamental por solucionar es determinar el número óptimo de clústers para separar “bien” un conjunto de datos. Numerosas aproximaciones se han hecho a lo largo de los años. Por ejemplo, en la literatura de la estadística se incluyen los índices Calinski y Harabasz (Calinski and Harabasz, 1974[26]), Dunn index (Dunn 1974[27]), la regla de Hartigan (1975[28]), silhouette width (Rousseeuw 1987[29]) y connectivity (Handl et al. 2005[30]).

- Calinski y Harabasz

$$G(u) = \frac{trace(Bu)/(u-1)}{trace(Wu)/(n-u)}$$

donde: $X = \{x_{ij}\}$, i=1, ..., n; j=1, …, m – matriz de datos.
n - número de objetos
m - número de variables
u - número de clústers (u= 2,...,n-1)

i) *Un algoritmo no determinista ofrece muchos posibles resultados con la misma entrada . No se puede saber de antemano cuál será el resultado de la ejecución de un algoritmo no determinístico.*
ii) *hierarchical clustering*

$Wu = \sum_r \sum_{i \in C_r} (x_{ri} - \bar{x}_r)(x_{ri} - \bar{x}_r)^T$ - dentro de la matriz de dispersión para los datos agrupados en clústers *u*

$Bu = \sum_r n_r (\bar{x}_r - \bar{x})(\bar{x}_r - \bar{x})^T$ - dentro de la matriz de dispersión para los datos agrupados en clústers *u*

$\bar{x}_r$ – *centroid o medoid del cluster r*
$\bar{x}$ – *centroid o medoid de la matriz de datos*
C_r – *índices de los objetos en el clúster r*
n_r – *número de objetos en el cluster r*

El valor de *u* que maximice $G_1(u)$ será el candidato para especificar el número de clústers que se usará para clasificar los datos.

- Hartigan

$$H(u) = \left(\frac{t_r W_u}{t_r W_{u+1}} - 1\right)(n - u - 1)$$

donde: $X=\{X_{ij}\}$, i= 1, … , n; j=1, … , m – data matrix
n – número de objetos del clúster
m – número de variables

$W_u = \sum_r \sum_{i \in C_r} (X_{ri} - \bar{X}_r)(X_{ri} - \bar{X}_r)^T$ - dentro de la matriz de dispersión para los datos agrupados en clústers *u*

X_{ri} – *vector de observaciones de m dimensiones del objeto i – ésimo en el clúster* r
$\bar{X}_r$ – *centroid o medoid del clúster* r
$r = 1, \ldots, u$ – *número de clúster*
u – *número de clústers* $(u = 1, \ldots, n-2)$
C_r – *los índices de los objetos dentro del clúster* r

El número óptimo de clústers es el más pequeño de u>=1 que H(u)<=10

- Dunn

El índice *Dunn* corresponde al ratio de la distancia más pequeña entre las observaciones de diferentes clústers y la distancia inter-cluster más grande ; se calcula del siguiente modo:

$$D(C) = \frac{\min\limits_{C_k, C_l \in C, C_k \neq C_l} \left(\min\limits_{i \in C_k, j \in C_l} dist(i, j) \right)}{\max\limits_{C_m \in C} diam(C_m)}$$

donde: C - conjunto de clúster

k, l, m - clústers

i, j - observaciones

dist(i,j) - distancia entre observaciones de diferentes clústers

diam(C_m) – máxima distancia entre las observaciones del clúster *m*

Tiene valores entre 0 e ∞ y debe maximizarse para que la agrupación sea la óptima. Mide cuan compactos (*fig. 2.16,b*) y separados (*fig. 2.16,c*) están los clústers entre ellos.

- Silhouette Width

El índice *silhouette width* , al igual que índice Dunn, mide cuan compactos y separados están los clústers. Este índice corresponde al promedio del valor silhouette de cada observación y mide el grado de confianza en la asignación de clústers de una observación en particular. Los valores próximos a 1 gozarán de una mayor confianza en la agrupación realizada, por el contrario, los valores próximos a -1 significan que la agrupación no es fiable. El intervalo de valores es [-1,1] . Para una observación *i* se define así:

$$S(i)=\frac{b_i-a_i}{max(b_i,a_i)}$$

donde: a_i – distancia promedio entre *i* y el resto de observaciones en el mismo clúster

B_i – distancia promedio entre *i* y el resto de observaciones del clúster vecino más cercano

- Connectivity

Por último, el índice *connectivity* (*fig. 2.16,a*) indica el grado de conectividad de los clústers y se determina a partir de los *L* vecinos más cercanos. Los valores de este índice van de 0 a ∞ y debe ser minimizado para obtener la agrupación óptima.

Definiendo $nn_{i(j)}$ como el j-ésimo vecino más próximo de la observación *i*, y que $X_i,nn_{i(j)}$ es 0 si *i* y nn_{ij} están en el mismo clúster y 1/j en otro caso. Entonces, para una agrupación particular de clústers $\{C=C_1,...,C_k\}$ con N observaciones agrupadas en K clústers disjuntos la conectividad se define de la siguiente manera:

$$Conn(C)=\sum_{i=1}^{N}\sum_{j=1}^{L} x_i, nn_{i(j)}$$

donde: L – determina el número de vecinos que contribuyen a la medida de conectividad.

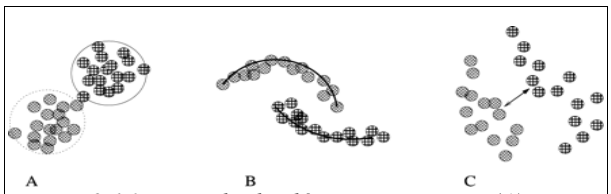


Figura 2.16: Ejemplo de clústers compactos (A), clústers con conectividad (B) y clústers con separación espacial (C)

2.2.4.- Intervalo de confianza

[31] Se llama *intervalo de confianza* en estadística a un par de números entre los cuales se estima que estará cierto valor desconocido con una determinada probabilidad de acierto. Formalmente, estos números determinan un intervalo, que se calcula a partir de datos de una muestra, y el valor desconocido es un parámetro poblacional.

La probabilidad de éxito en la estimación se representa por *1 - α* y se denomina nivel de confianza. En estas circunstancias, α es el llamado error aleatorio o nivel de significación, esto es, una medida de las posibilidades de fallar en la estimación mediante tal intervalo.

El nivel de confianza y la amplitud del intervalo varían conjuntamente, de forma que un intervalo más amplio tendrá más posibilidades de acierto (mayor nivel de confianza), mientras que para un intervalo más pequeño, que ofrece una estimación más precisa, aumentan sus posibilidades de error.

Para la construcción de un determinado intervalo de confianza es necesario conocer la distribución teórica que sigue el parámetro a estimar, *θ*. Es habitual que el parámetro se distribuya normalmente.

En definitiva, un intervalo de confianza al 1 - α% para la estimación de un parámetro poblacional *θ* que sigue una determinada distribución de probabilidad, es una expresión del tipo $[\theta_1, \theta_2]$ tal que $P[\theta_1 \leq \theta \leq \theta_2] = 1 - \alpha$, donde *P* es la función de distribución de probabilidad de *θ*.

De una población de media μ y desviación típica σ se pueden tomar condiciones muestrales de *n* elementos. Cada una de estas condiciones muestrales tiene a su vez una media ($\bar{x}$). Se puede demostrar que la media de todas las medias muestrales coincide con la media poblacional: $\mu_{\bar{x}}=\mu$

Pero además, si el tamaño de las condiciones muestrales es lo suficientemente grande, la distribución de medias muestrales es, prácticamente, una distribución normal (o gaussiana) con media μ y

una desviación típica dada por la siguiente expresión: $\sigma_{\bar{x}}=\frac{\sigma}{\sqrt{n}}$

En una distribución normal puede calcularse fácilmente un intervalo dentro del cual "caigan" un determinado porcentaje de las observaciones, esto es, es sencillo hallar z_1 y z_2 tales que $P[z_1 \leq z \leq z_2] = 1 - \alpha$, donde (1 - α)*100 es el porcentaje deseado .

Se desea obtener una expresión tal que $P[\mu_1 \leqslant \mu \leqslant \mu_2]=1-\alpha$.

En esta distribución normal de medias se puede calcular el intervalo de confianza donde se encontrará la media poblacional si sólo se conoce una media muestral ($\bar{x}$), con una confianza determinada. Habitualmente se manejan valores de confianza del 95% y 99%. A este valor se le llamará *1 − α* (debido a que α es el error que se cometerá, un término opuesto).

Para ello se necesita calcular el punto $X_{\alpha/2}$ — o mejor dicho su versión estandarizada $Z_{\alpha/2}$— junto con su "opuesto en la distribución" $X_{-\alpha/2}$. Estos puntos delimitan la probabilidad para el intervalo, como se muestra en la *figura 2.17*

El intervalo se define del siguiente modo: $(\bar{x}-z_{\alpha/2}\frac{\sigma}{\sqrt{n}}, \bar{x}+z_{\alpha/2}\frac{\sigma}{\sqrt{n}})$ donde σ corresponde a la desviación típica de la distribución, $\bar{x}$ es la media de población , *n* el número de condiciones muestrales de la población , $Z_{\alpha/2}$ constante de la distribución según probabilidad de confianza 1- α y σ/√n como el error estándar . Existen unas tablas precalculadas para las distribuciones normales que definen esta constante para una determinada probabilidad de confianza (1-α), también existen las tablas a la inversa[31] .

[32]La distribución t de Student es una distribución de probabilidad que surge del problema de estimar la media de una población normalmente distribuida cuando el tamaño de la muestra es pequeño.

Aparece de manera natural al realizar la prueba t de Student para la determinación de las diferencias entre dos medias muestrales y para la construcción del intervalo de confianza para la diferencia entre las medias de dos poblaciones cuando se desconoce la desviación típica de una población y ésta debe ser estimada a partir de los datos de una muestra.

El procedimiento para el cálculo del intervalo de confianza basado en la t de Student consiste en estimar la desviación típica de los datos S, el número de condiciones muestrales *n* y calcular el error estándar de la media= $\frac{S}{\sqrt{n}}$, siendo entonces el intervalo de confianza para la media = $\bar{x} \pm t(\frac{\alpha}{2})(\frac{S}{\sqrt{n}})$ donde t($\frac{\alpha}{2}$) es una constante numérica dependiente de la probabilidad de confianza (1-α). Esta constante se puede obtener a partir de unas tablas ya precalculadas para la distribución t de Student.

Es este resultado el que se utiliza en el test de Student; puesto que la diferencia de las medias de condiciones muestrales de dos distribuciones normales se distribuye también normalmente, la distribución *t* puede usarse para examinar si esa diferencia puede razonablemente suponerse igual a cero.

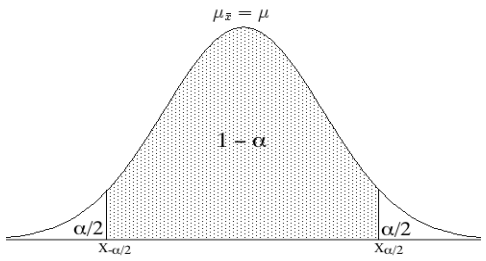


Figura 2.17: Ejemplo de intervalo de confianza para una distribución normal. 1-α corresponde a la probabilidad de que una muestra esté dentro del intervalo (nivel de confianza). Con los dos puntos del intervalo se definen las áreas de error. La suma de las dos áreas corresponde a α , que es el error que se comete (error de la distribución normal).

3.- Fases

A continuación se muestra una tabla comparativa entre la planificación inicial y la planificación final del trabajo desarrollado.

PLANIFICACIÓN INICIAL	PLANIFICACIÓN FINAL
Formación	
Estudio del paquete estadístico CLUTO y R-STATISTICS para elegir uno de ellos	
Estudio e implementación algoritmos de agrupación con la herramienta escogida	
Estudio de los distintos índices de integridad escogidos	
Búsqueda de la agrupación óptima a partir de los índices de integridad escogidos	Búsqueda de la mejor agrupación para cada uno de los algoritmos de agrupación según los índices de integridad escogidos
	Tratamiento de condiciones muestrales outlayers. Agrupación con fusión de condiciones muestrales outlayers a otros clústers y agrupación sin fusión pero con outlayers**(mejora).**
Integración de la agrupación en el preproceso	
Gestión de una Base de Datos para guardar los resultados y guardar datos del histórico de los usuarios	Gestión de ficheros en vez de usar base de datos para almacenar la información resultante y para gestionar el histórico de los usuarios
Integración de la agrupación en el aplicativo web PCOPGene	
	Normalización y agrupación de las mejores agrupaciones por similitud e integración en PCOPGene (**mejora**)
Búsqueda de genes marcadores basada en los intervalos de confianza	
Integración de la búsqueda en el aplicativo web PCOPGene	

Para las pruebas se ha usado una microarray proporcionada por la base de datos del Instituto Nacional del Cáncer (EE.UU.) (Scherf et al 2000)[35] . Corresponde a los perfiles de 9.703 cDNAs que representa ~ 8000 genes únicos de 60 líneas celulares, en relación con los perfiles de actividad de 1400 medicamentos (finalmente resumidos en 1417 genes de 40 lineas celulares, en relación a 118 medicamentos, que representan los valores más significativos). A esta microarray se la identifica como *m17.*

En los siguientes apartados se expone la planificación de las distintas fases, los problemas encontrados y como se han solucionado éstos satisfactoriamente.

3.1.- Formación

Para el desarrollo de la aplicación se han tenido que adquirir conocimientos básicos de biología molecular.

También ha sido necesaria una formación básica en los algoritmos de agrupación más usados y en algunos de los índices de integridad que miden cuan óptimo es el resultado.

Para implementar la búsqueda de genes marcadores se requieren conocimientos básicos de estadística, concretamente de las distribuciones normales e intervalos de confianza.

Para realizar la implementación se ha tenido que aprender a utilizar la herramienta web PCOPGene [14] leyendo manuales de ayuda de dicha herramienta.

A parte de la base teórica se ha realizado un aprendizaje de la herramienta R-STATISTICS[2] que es la que se usa para el desarrollo de la aplicación.

Finalmente se han tenido que refrescar conocimientos de programación PHP[i], PERL[ii] y C[iii].

3.2.- Paquetes Estadísticos

El primer paso de todos consiste en encontrar una herramienta capaz de realizar todos los cálculos requeridos y que además cumpla con una serie de requisitos respecto a la aplicación que se desarrolla.

Las dos candidatas son CLUTO[1] y R-Statistics[2]. Estas dos herramientas son las más utilizadas en el análisis genómico.

A pesar de que ambas herramientas son paquetes muy extendidos se ha optado por usar R-Statistics (R) por tres motivos:

– R es un lenguaje y entorno de programación open source[iv] que lo hace mucho más flexible y versátil que CLUTO

– R ofrece librerías que permiten integrar de manera muy sencilla procesos realizados en R dentro de procesos desarrollados en C o Perl, en CLUTO es más difícil este proceso

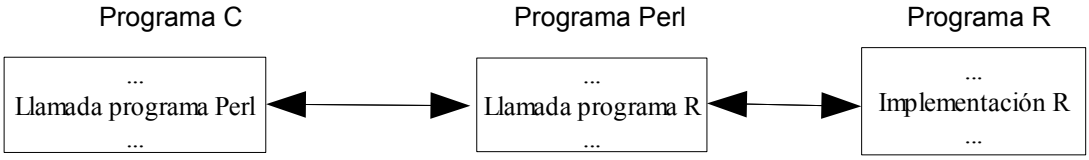
– R es un paquete estadístico mucho mayor que CLUTO puesto que evoluciona constantemente gracias a la colaboración de investigadores que desarrollan sus propias librerías y después las incorporan al repositorio[v] de R (Comprehensive R Archive Network,CRAN[3])

Una vez elegido R como paquete estadístico para la implementación de los diferentes algoritmos de clustering se procede al estudio de como integrar operaciones de R en C

3.3.- Integración de R-Statistic en lenguaje C

R-Statistics(R) ofrece la posibilidad de implementar operaciones R en lenguaje C mediante sus propias librerías R. Su utilización está más orientado al desarrollo de nuevas librerías para R y no tanto para la implementación de operaciones de R en C. A parte, la utilización de este método de implementación requiere de un alto conocimiento en programación puesto que no es una tarea fácil su integración en C.

Dado que R ofrece un par de librerías para integrar R en Perl de manera sencilla e intuitiva se opta por analizar estas dos librerías para poder decantarnos por una de ellas. Como Perl tiene características del lenguaje C, del lenguaje interpretado shell (sh), AWK, sed, Lisp y, en un grado inferior, de muchos otros lenguajes de programación, es muy fácil de integrar implementaciones Perl en C. Por lo tanto, este es el modelo de implementación:



Estas dos librerías de R para Perl son RSPerl[33] y Statistics::R[34].

La finalidad de ambas librerías es la misma, inicializar el intérprete[vi] R y poder mandar operaciones R desde Perl a través de un objeto de interconexión entre Perl y el intérprete R. Todo ello se realiza con

i) *PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas .*
ii) *Perl es un lenguaje de programación diseñado por Larry Wall en 1987. Perl toma características del lenguaje C, del lenguaje interpretado shell (sh), AWK, sed, Lisp y, en un grado inferior, de muchos otros lenguajes de programación.*
iii) *C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell*
iv) *Open Source (Código abierto) es el término con el que se conoce al software distribuido y desarrollado libremente*
v) *Un repositorio es un sitio centralizado donde se almacena y mantiene información digital.*
vi) *Un intérprete o interpretador es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel*

métodos que ofrecen estas librerías.

Las primeras pruebas se realizan con RSPerl pero un error de incompatibilidad con el sistema hace que nos decantemos por Statistics::R sin perder ninguna funcionalidad.

Una vez escogida la librería se pasa a analizar e implementar los diferentes algoritmos de agrupación.

3.4.- Agrupación de condiciones muestrales

Para implementar las operaciones de R primero se testean directamente en el intérprete de R y una vez está todo correcto se implementa el script[i] de Perl que realizará las llamadas a estas operaciones.

El primer paso de todos consiste en realizar la lectura desde R de los datos de las microarrays.

Una vez leídos los datos y almacenados se deben rellenar los posibles valores vacíos que haya en la microarray.

Cuando se hayan adquirido los datos de entrada ya se pueden implementar los diferentes algoritmos de agrupación así como su validación mediante los cálculos de integridad.

3.4.1.- Lectura de la microarray de entrada

Las microarrays son matrices numéricas donde las filas corresponden a cada uno de los genes analizados y las columnas son las condiciones muestrales a las que se han sometido estos genes. Los genes son las los individuos y las condiciones las variables.

En nuestro caso interesa realizar el estudio sobre las condiciones muestrales, es decir, las condiciones serán los individuos y los genes las variables. Por lo tanto, tendremos que obtener la matriz transpuesta[ii] de la microarray de entrada.

3.4.2.- Corrección de “celdas vacías” en la microarray de entrada

En el proceso de obtención de la matriz numérica de los niveles de expresión de unos determinados genes bajo unas determinadas condiciones muestrales es posible que ocurran errores. Estos errores se traducen en hueco o valores vacíos en la matriz numérica.

Para corregir estos errores se escoge una librería de R-Statistics que se encarga de rellenar los huecos automáticamente.

A continuación se muestra un ejemplo de como funciona este método.

En la matriz que se muestra en la *figura 3.1* se puede observar como el gen 2 tiene un elemento vacío para la condición 10. Para este ejemplo se buscan los 2 vecinos más próximos al gen 2. Primero de todo se calculan las distancias euclidianas entre el gen 2 y el resto de genes limitando las columnas de la condición 1 a la condición 8 *(p.e* $dist(Gen_{2,}Gen_1)=\sqrt{((cond1_{gen2}-cond1_{gen1})^2+...+(cond8_{gen2}-cond8_{gen1})^2)}$). Los genes 1 y 4 son los vecinos más próximos al gen 2 porque tiene las distancias más pequeñas respecto al gen 2. Una vez encontrado los 2 vecinos calculamos el valor promedio de los elementos correspondientes a la condición 10 de estos vecinos, en este caso -0.185. Este será el valor que se usará para rellenar el elemento vacío del Gen 2 para la condición 10.

i) *Un **script** (cuya traducción literal es 'guión') o archivo de órdenes o archivo de procesamiento por lotes es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Son interpretados.*

ii) *Sea A una matriz con m filas y n columnas. La matriz traspuesta, denotada con* A^t *está dada por:* $A^t_{(i,j)}=A_{ji}, 1\leq i \leq n, 1\leq j\leq m$

	Cond_1	Cond_2	Cond_3	Cond_4	Cond_5	Cond_6	Cond_7	Cond_8	Cond_10
Gen_1	**-0.2**	**-0.19**	**-0.14**	**-0.08**	**-0.16**	**-0.41**	**-0.21**	**-0.3**	**-0.2**
Gen_2	-0.09	-0.14	-0.18	-0.11	-0.2	-0.12	-0.38	-0.22	**0/ -0.185**
Gen_3	-0.36	-0.16	-0.19	-0.13	-0.29	-0.12	-0.02	-0.22	-0.12
Gen_4	**-0.14**	**-0.06**	**-0.07**	**-0.21**	**-0.3**	**-0.15**	**-0.19**	**-0.38**	**-0.17**
Gen_5	-0.19	0.11	0.03	0.02	-0.08	-0.12	-0.1	-0.23	-0.03
Gen_6	-0.19	-0.16	-0.1	-0.19	-0.17	-0.35	-0.05	-0.19	-0.08

Figura 3.1: Ejemplo de matriz con valores vacíos. En azul se marca el elemento vacío que corresponde a la condición muestral del gen 2. En rojo se marca los 2 vecinos más próximos al gen 2 usando la métrica euclidiana como medida de distancia. A partir de estos vecinos se calcula el valor que se usa para rellenar el elemento vacío. En este caso se hace el promedio de (-0.2 y -0.17) que corresponden a la condición 10 de los vecinos más próximos. Este valor es el que está en verde, -0.185.

Para favorecer la eficiencia del algoritmo es recomendable ordenar los genes por vecindad.

3.4.3.- Agrupación de las condiciones muestrales: PC, MDS, SOM ,SOTA, PAM, HC, K-MEANS

Los diferentes algoritmos de agrupación que se implementan son :SOM (Self-organizing Maps), SOTA (Self-organizing Tree Algorithm), PAM (Partitioning Around Medoids) y HC (hierarchical clustering) y K-MEANS.

Escalando la microarray de entrada obtenemos dos nuevas matrices de datos :PC (Principal Components) y MDS(Multidimensional Scaling).

El escalado PC facilita el estudio de los clústers encontrados puesto que reduce las variables de la microarray de entrada a 2 y permite representar los clústers en un plano 2D como se muestra en la *fig. 3.2*.

A las matrices escaladas se les aplican todos los algoritmos de agrupación; a la microarray de entrada también se le aplican las mismas menos el K-MEANS.

Inicialmente se buscaron librerías individuales para cada uno de los algoritmos de agrupación hasta que se encontró una única librería que realizaba todos los algoritmos de agrupación requeridos a la vez que realizaba cálculos de integridad sobre las agrupaciones obtenidas. Esta librería se llama *clValid*[11].

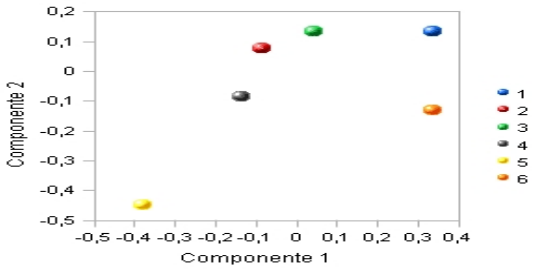


Figura 3.2: Representación 2D de una matriz de dos componentes principales calculada a partir de una de 6 variables usando PC. La muestra 5 sería un outlayer (sin clúster) y el resto podrían configurar un clúster

3.4.4.- Cálculo de la Integridad de las distribuciones de clústers

Inicialmente el objetivo marcado era encontrar una única agrupación óptima para cada uno de los diferentes métodos de agrupación. Para ello se tendrían que combinar los diferentes índices de integridad.

Después de realizar un estudio exhaustivo de los resultados se llegó a la conclusión de que combinando los índices de integridad no se podía encontrar una única agrupación óptima así que se optó por guardar la agrupación óptima para cada índice de integridad, es decir, cada algoritmo de agrupación

podría tener más de una agrupación (una por cada índice de integridad, la más óptima). Todos estos análisis y estudios se explican a continuación.

Se utilizó la microarray *m17* porque tanto Mario Huerta como Juan Antonio Cedano habían realizado varios estudios sobre ella y ya tenían realizada una agrupación basada en los algoritmos *PC y MDS* que se podría utilizar para comparar con los futuros resultados obtenidos con los diferentes algoritmos de agrupación. A esta agrupación la llamamos *default colors.*

Se realizaron tres estudios, uno sobre los índices Calinski-Harabasz y Hartigan, el otro sobre los índices Connectivity, Silhouette y Dunn y finalmente una validación visual de las agrupaciones obtenidas.

·Calinski-Harabasz y Hartigan

Para estudiar la fiabilidad de estos dos índices de integridad se analizan los resultados obtenidos de aplicar la agrupación *som* a la microarray *m17* para el rango de *k's*[2,8]

En la *tabla 3.1* se muestran los resultados obtenidos. Para cada combinación x/y del mapa bidimensional tales que $x*y \in [2..8]$. Se puede observar que la mejor *k* que optimiza la agrupación para los dos índices es 2. No es un buen resultado ya que dividir el conjunto de condiciones muestrales en tan pocos clústers no permite distinguir los distintos grupos de estados celulares que puedan existir dentro de estas condiciones muestrales.

Analizando la *tabla 3.1* también se puede observar como los valores para el índice *Calinski-Harabasz* decrecen a la vez que se incrementa el número de clústers. Por lo tanto, la mejor *k* siempre será la *k* menor de todo el estudio (K=2 en este caso). No interesa agrupar las condiciones muestrales para k's pequeñas por la razón que se explicó antes. Por este motivo se descarta este índice de integridad.

Para validar que las agrupaciones obtenidas a partir del algoritmo *som* eran correctos se optó por usar una de las funcionalidades que tiene la herramienta PCOPGene[14]. Esta herramienta se llama *depend.analysis* .

A nivel visual *depend.analysis* genera una gráfica 2D disponiendo todas las condiciones muestrales de la microarray en el plano usando la relación entre dos genes previamente seleccionados. Cada clúster de condiciones muestrales tiene su propio color y las condiciones muestrales se pintan con el color del clúster al que pertenece. Existe un fichero *colors (*plantilla colors*)* que contiene las asignaciones de clústers de cada muestra. Esta plantilla colors se genera junto con la agrupación.

Para la validación se compararon dos plantillas de colores (agrupaciones) diferentes. Una era la *plantilla default* de 9 clústers (el clúster 9 no se tiene en cuenta, son condiciones muestrales outlayers) y la otra se obtuvo a partir de la agrupación generado por la librería *som* de 8 clústers. Para ambas plantillas se usaron los genes QSOX1 y GUCY1B3.

En la *fig. 3.3* tenemos la relación entre las expresiones de estos dos genes coloreando los clústers usando la *plantilla default*.

En la *fig. 3.4* se representa también la relación entre estos dos genes usando la plantilla generada a partir de la agrupación *som* con xDim=2 y yDim=4.

Comparando las *figuras 3.3* y *3.4* se puede ver como los grupos de condiciones muestrales obtenidos es muy similar con pequeñas variaciones por lo tanto podemos deducir que la agrupación de condiciones muestrales obtenida a partir de la librería *som* es una buena aproximación.

CÁLCULO INTEGRIDADES CLUSTERING SOM (librería som)			
X-Y	K	Calinski-Harabasz	Hartigan Index
1-2	2	45.29525	9.00101
1-3	3	28.60326	2.954459
1-4	4	20.35231	3.190721
2-2	4	30.83795	-14.81250
1-5	5	16.33394	0.9610587
1-6	6	13.24954	1.592263
2-3	6	23.37993	-23.76523
1-7	7	11.35757	0.9146093
1-8	8	9.854467	2.748154
2-4	8	17.03204	-24.32977

Tabla 3.1: Valores integridad para la agrupación som de la microarray m17 para k=2..8 Según el índice de Calinski la k óptima sería 2 con valor 45,3 puesto que es la k que maximiza este índice. Para el índice Hartigan la k óptima sería también 2 ya que es la k más pequeña con valor por debajo de 10 (9,001). La primera columna corresponde a la dimensionalidad de los dos ejes, X e Y, la segunda columna es el número de clústers en que se agrupan las condiciones muestrales y las dos columnas restantes son los valores de los índices de integridad.

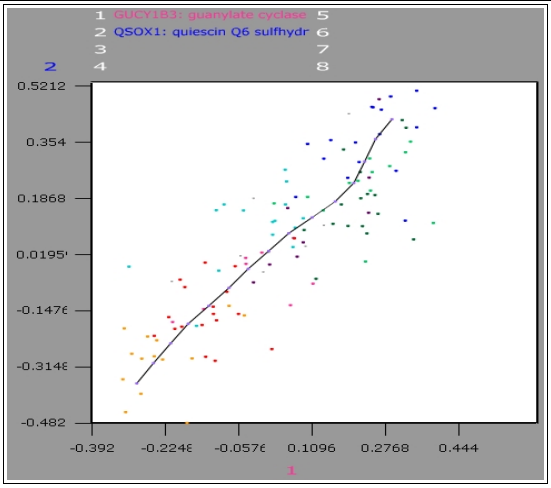


Figura 3.3: Relación niveles de expresión entre los genes GUCYB3 (eje x) y QSOX1 (eje y) usando la agrupación de 9 clústers optimizada por Mario Huerta y Juan Antonio Cedano. Cada color representa un clúster distinto. Las condiciones muestrales de color gris son outlayers que pertenecen al clúster 9. Los outlayers no se tienen en cuenta por lo tanto tenemos realmente 8 clústers descartando los outlayers

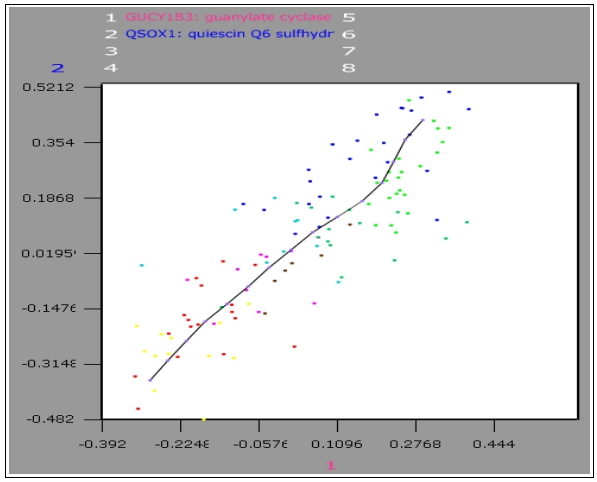


Figura 3.4: Relación niveles de expresión entre los genes GUCYB3(eje x) y QSOX1(eje y) usando la agrupación de 8 clústers obtenida a partir de la librería som. Cada color representa un clúster distinto. En esta agrupación los outlayers que aparecen en la fig. 3.3 se han agrupado en otros clústers teniendo al final una agrupación de 8 clústers.

Tanto en el estudio anterior como en el que se explicará a continuación se permitieron agrupaciones que contenían condiciones muestrales del tipo outlayers[i]. Una agrupación con este tipo de condiciones muestrales puede conducir a errores en el análisis de los grupos de condiciones muestrales. Lo ideal es trabajar con agrupaciones que no contengan outlayer, (*sección 3.5*)

·Connectivity, Silhouette y Dunn

Para analizar estos índices de integridad se tuvo que analizar también la métrica que usa la librería *clValid* para el cálculo de la agrupación.

i) *Un outlayer es una muestra que no tiene grupo asignado y por lo tanto no se tienen en cuenta en los estudios. Los clústers deben tener un número mínimo de condiciones muestrales, todas las condiciones muestrales que pertenezcan a un clúster que no cumpla esta condición se les llama outlayers.*

Se analizaron los resultados obtenidos a partir de la agrupación *(clValid)* de la microarray *m17* escalada usando PC para cada una de las métricas *Euclidean, Manhattan y Correlation*.

Las tablas *3.2, 3.3 y 3.4* contienen los resultados obtenidos usando las métricas *Euclidean, Manhattan y Correlation* respectivamente.

En las tablas *3.2 y 3.3 (*métricas *euclidean y manhattan)* observamos como para el índice *silhouette* la mejor *k* para la mayoría de los algoritmos es 2. En cambio para el índice D*unn* no es posible definir la mejor *k* para la mayoría de algoritmos.

En la tabla *3.4* (métrica *Correlation*) se puede observar como para el índice *Dunn* no se obtienen valores. Esta es la razón principal por la que se descarta este tipo de métrica ya que no nos permitiría usar el índice Dunn.

Visto que los resultados de los índices de integridad para las métricas *Euclidiana y Manhattan* son muy similares se opta por usar la métrica *euclidiana* que es la métrica por defecto que usa *clvalid*.

Estudiando las gráficas de las *fig. 3.5, 3.6,3.7* se discriminan alguno de los 3 índices de integridad.

Cada gráfica corresponde a la relación entre un índice de integridad determinado y el número de clústers de cada unos de los algoritmos de agrupación calculados usando la métrica *euclidean* (más resultados en el Anexo 3).

Recordar que para estos índices de integridad la mejor *k* es aquella que maximiza su valor.

Como se muestra el la gráfica de la *fig.3.5* el índice *Connectivity* tiene una evolución ascendente. Es decir la mejor *k* siempre será la mayor de todas las calculadas. Este resultado no es bueno ya que no siempre la *k* mayor es la mejor, cuantos más clústers más riesgo de la aparición de *outlayers* y de clústers con pocas condiciones muestrales. La idea es agrupar las condiciones muestrales en los menos clústers posibles, con una cantidad de condiciones muestrales significativas en cada uno de los clústers y sobretodo que aporten información sobre posibles estados celulares.

La evolución del índice *Silhouette* (*fig.3.6*) *y Dunn* (*fig.3.7*) es una evolución discontinua.

Para el índice *Silhouette* (*fig.3.6)* se puede definir la mejor *k* para la mayoría de algoritmos (k=2) .

En el caso del índice *Dunn* (*fig.3.7)* no se puede definir la mejor *k* para la mayoría de algoritmos.

Por este motivo se opta por escoger la mejor *k* para cada uno de los algortimos implementados y dejar al usuario que escoja que escoja de entre éstos.

Después de todos estos análisis se descartaron los índices de integridad de *Calinski-Harabasz* y *Connectivity* y se escogieron los índices de *Hartigan, Dunn* y *Silhouette.* La métrica escogida fue la E*uclidiana.*

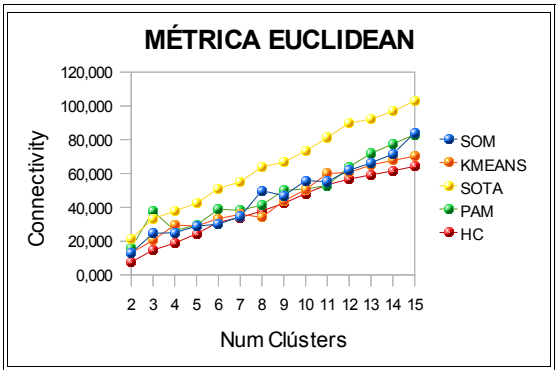


Figura 3.5: Para cada uno de los algoritmos de agrupación (series) representamos su valor de integridad connectivity (eje Y) para cada número de clústers(eje X). Los datos que se usan para la agrupación corresponden a la microarray escalada usando PC. La métrica que se usa es la euclidiana. Se observa como la k óptima siempre es la mayor, o sea, 15 para todas las agrupaciones.

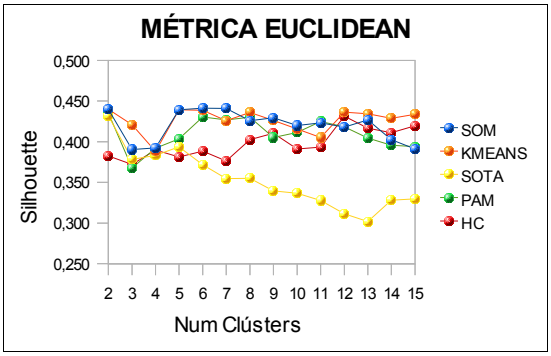


Figura 3.6:Para cada uno de los algoritmos de agrupación (series) representamos su valor de integridad silhouette (eje Y) para cada número de clústers(eje X). Los datos que se usan para la agrupación corresponden a la microarray escalada usando PC. La métrica que se usa es la euclidiana. La k óptima para todos los algoritmos es 2 excepto para SOM y HC que es 6 y 15 respectivamente

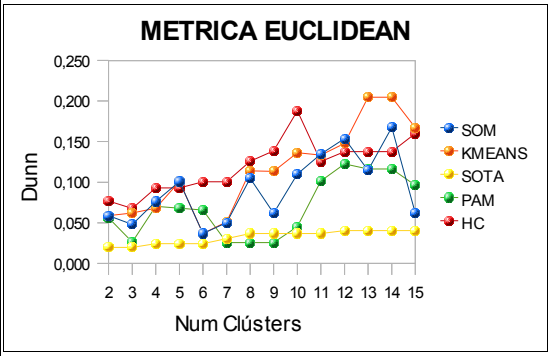


Figura 3.7: Para cada uno de los algoritmos de agrupación (series) representamos su valor de integridad dunn (eje Y) para cada número de clústers(eje X). Los datos que se usan para la agrupación corresponden a la microarray escalada usando PC. La métrica que se usa es la euclidiana. En este caso no se puede encontrar una K óptima para la mayoría de algoritmos.

EUCLIDEAN

NumClusters	Dunn KMEANS	Dunn SOTA	Dunn SOM	Dunn PAM	Dunn HC	Silhouette KMEANS	Silhouette SOTA	Silhouette SOM	Silhouette PAM	Silhouette HC
2	0,059	0,019773	0,059	0,056	0,077	0,440	0,432	0,440	0,435	0,383
3	0,062	0,019773	0,048	0,026	0,068	0,421	0,378	0,390	0,368	0,372
4	0,068	0,024445	0,076	0,070	0,093	0,388	0,383	0,392	0,392	0,390
5	0,101	0,024445	0,101	0,068	0,093	0,439	0,394	0,439	0,404	0,381
6	0,037	0,024445	0,037	0,066	0,101	0,439	0,372	0,441	0,431	0,389
7	0,051	0,030602	0,050	0,026	0,101	0,426	0,355	0,441	0,427	0,376
8	0,114	0,036903	0,105	0,026	0,126	0,436	0,355	0,426	0,430	0,403
9	0,113	0,036903	0,062	0,026	0,139	0,426	0,339	0,430	0,405	0,411
10	0,136	0,036903	0,110	0,045	0,188	0,416	0,337	0,420	0,411	0,391
11	0,133	0,036903	0,135	0,101	0,125	0,405	0,328	0,423	0,425	0,393
12	0,148	0,040412	0,154	0,123	0,138	0,436	0,311	0,418	0,418	0,431
13	0,205	0,040412	0,115	0,116	0,138	0,434	0,302	0,427	0,405	0,417
14	0,205	0,040412	0,168	0,116	0,138	0,429	0,328	0,403	0,396	0,411
15	0,167	0,040412	0,062	0,096	0,160	0,434	0,330	0,391	0,394	0,420

Tabla 3.2: Valores de los índices de integridad Dunn y Silhouette para cada uno de los algoritmos de agrupación calculados a partir de la agrupación de las condiciones muestrales de la microarray escalada a dos dimensiones usando PC y usando como métrica la distancia Euclidiana. Se resalta en amarillo las k (número de clústers) óptimas . Se observa como con el índice Silhouette se puede concretar una k óptima(2) para la mayoría de métodos de agrupaciones , en cambio para el índice Dunn no se puede definir una k óptima.

MANHATTAN

NumClusters	Dunn KMEANS	Dunn SOTA	Dunn SOM	Dunn PAM	Dunn HC	Silhouette KMEANS	Silhouette SOTA	Silhouette SOM	Silhouette PAM	Silhouette HC
2	0,042	0,017	0,038	0,083	0,067	0,440	0,435	0,436	0,434	0,402
3	0,051	0,017	0,047	0,062	0,072	0,412	0,371	0,410	0,356	0,376
4	0,050	0,025	0,057	0,029	0,075	0,389	0,382	0,398	0,334	0,377
5	0,104	0,025	0,070	0,039	0,075	0,424	0,385	0,384	0,372	0,382
6	0,082	0,026	0,104	0,069	0,108	0,428	0,361	0,401	0,415	0,403
7	0,043	0,027	0,082	0,110	0,108	0,413	0,342	0,408	0,421	0,391
8	0,118	0,033	0,054	0,110	0,116	0,423	0,357	0,405	0,425	0,413
9	0,108	0,033	0,104	0,127	0,116	0,427	0,338	0,432	0,409	0,396
10	0,122	0,033	0,094	0,045	0,132	0,420	0,329	0,403	0,413	0,373
11	0,117	0,033	0,150	0,103	0,132	0,394	0,324	0,427	0,430	0,397
12	0,102	0,038	0,084	0,107	0,150	0,386	0,308	0,412	0,431	0,382
13	0,102	0,038	0,058	0,107	0,150	0,390	0,307	0,390	0,406	0,381
14	0,107	0,038	0,036	0,094	0,180	0,427	0,331	0,375	0,405	0,421
15	0,128	0,038	0,117	0,111	0,186	0,434	0,330	0,380	0,397	0,408

Tabla 3.3:Valores de los índices de integridad Dunn y Silhouette para cada uno de los algoritmos de agrupación calculados a partir del agrupamiento de las condiciones muestrales de la microarray escalada a dos dimensiones usando PC y usando como métrica la distancia Manhattan. Se resalta en amarillo las k (número de clústers) óptimas . Se observa como con el índice Silhouette se puede concretar una k óptima(2) para la mayoría de métodos de agrupaciones , en cambio para el índice Dunn no se puede definir una k óptima.

CORRELATION										
NumClusters	Dunn KMEANS	Dunn SOTA	Dunn SOM	Dunn PAM	Dunn HC	Silhouette KMEANS	Silhouette SOTA	Silhouette SOM	Silhouette PAM	Silhouette HC
2	0,000	0,000	0,000	6,00E+015	6,00E+015	0,484	0,635	0,484	1,000	1,000
3	0,000	0,000	0,000	0,000	0,000	0,417	0,363	0,446	0,333	0,626
4	0,000	0,000	0,000	0,000	0,000	0,017	0,203	0,153	0,142	0,221
5	0,000	0,000	0,000	0,000	0,000	-0,126	-0,105	-0,119	0,074	0,106
6	0,000	0,000	0,000	0,000	0,000	-0,273	-0,030	-0,341	0,054	0,057
7	0,000	0,000	0,000	0,000	0,000	-0,298	-0,030	-0,327	0,052	0,015
8	0,000	0,000	0,000	0,000	0,000	-0,391	-0,351	-0,375	0,047	-0,114
9	0,000	0,000	0,000	0,000	0,000	-0,418	-0,351	-0,498	0,065	-0,210
10	0,000	0,000	0,000	0,000	0,000	-0,319	-0,399	-0,523	-0,722	-0,229
11	0,000	0,000	0,000	0,000	0,000	-0,469	-0,421	-0,510	-0,713	-0,214
12	0,000	0,000	0,000	0,000	0,000	-0,421	-0,492	-0,440	-0,724	-0,234
13	0,000	0,000	0,000	0,000	0,000	-0,463	-0,480	-0,462	-0,730	-0,219
14	0,000	0,000	0,000	0,000	0,000	-0,474	-0,509	-0,455	-0,721	-0,217
15	0,000	0,000	0,000	0,000	0,000	-0,528	-0,511	-0,521	-0,802	-0,232

Tabla 3.4: Valores de los índices de integridad Dunn y Silhouette para cada uno de los algoritmos de agrupación calculados a partir del agrupamiento de las condiciones muestrales de la microarray escalada a dos dimensiones usando PC y usando como métrica la Correlación. Se resalta en amarillo las k (número de clústers) óptimas . Se observa como con el índice Silhouette se puede concretar una k óptima(2) para todos los algoritmos pero el índice Dunn da valores 0 . Por esta razón se descarta el uso de esta métrica.

Estudiados los índices de integridad y la métrica se pasó a validar visualmente los grupos de condiciones muestrales que se obtuvieron para cada algoritmo y para cada *k*.

·Validación visual de las distribuciones de clústers

En la *tabla 3.5* tenemos por ejemplo la agrupación obtenida a partir del algoritmo PAM de 8 clústers. En las gráficas cada uno de estos clústers tendrá asignado un color distinto. Por este motivo se le llama plantilla de colores.

La plantilla que se usó para compara las agrupaciones resultantes de *clvalid* fue la *plantilla default*[i].

En la *fig. 3.8* se muestran los grupos de condiciones muestrales formados a partir de esta agrupación identificando cada grupo con un color distinto. El grupo outlayers corresponde a condiciones muestrales que no han sido asignadas a ningún grupo y que no aportan mucha información en este caso.

En las figuras *3.9, 3.10 y 3.11* se representan los clústers obtenidos a partir de los algoritmos *PAM*, *SOM y SOTA* respectivamente. En estas gráficas se muestra el resultado para la agrupación de 8 clústers.

Las condiciones muestrales se disponen en el plano usando como coordenadas *x* e *y* la componente1 y componente 2 respectivamente. A cada clúster se le asignó un color distinto.

Si se comparan los distintos clústers de cada una de las agrupaciones con los clústers de la *plantilla default* se puede observar que el resultado es muy similar, es más, en la plantilla *default* aparecen condiciones muestrales que no pertenecen a ningún clúster (*outlayers*) cosa que en el resto de agrupaciones no sucede (se han asignado a algún clúster cercano).

Por lo tanto podemos afirmar que los algoritmos funcionan correctamente y no hacen agrupaciones extrañas que es lo que interesaba observar en este tipo de análisis.

Un último análisis consistió en estudiar si se obtenían mejores resultados realizando la agrupación sobre la microarray directamente o sobre la microarray escalada.

Para realizar esta tarea se compararon dos plantillas, una la obtenida a partir de aplicar el algoritmo *som* directamente sobre los datos(*fig. 3.12*) y la otra obtenida al aplicar este mismo algoritmo pero sobre los datos escalados mediante *PC* (*3.10*).

Se puede observar como en la *fig.3.10* (*PC-SOM*) los clústers están mejor definidos, no es el caso para la *fig. 3.12* (*SOM*) en la cual se puede observar como por ejemplo los clústers 6 y 5 se solapan. A parte en la *fig. 3.12*(*SOM*) aparecen una serie de condiciones muestrales (color negro) que no aportan información y no tienen un clúster definido (outlayers) cosa que no ocurre con el método *PC-SOM.*

Llegamos a la conclusión de que aplicando el escalado previo de la microarray de entrada usando *PC* y después aplicando los distintos algoritmos de agrupación obtenemos una mejor agrupación de las condiciones muestrales comparado con aplicar directamente a la microarray los distintos algoritmos de agrupación.

i) agrupación de condiciones muestrales de la matriz m17 basado en los algoritmos PC y MDS optimizado por Mario Huerta y J.A. Cedano

ID CLÚSTER	ID MUESTRA	ID CLÚSTER	ID MUESTRA	ID CLÚSTER	ID MUESTRA
1	1	4	24	7	46
1	3	4	40	7	47
1	4	4	43	7	49
1	5	4	44	7	50
1	6	4	45	7	53
1	10	4	51	7	56
1	41	4	52	7	57
1	42	4	54	7	58
1	81	4	55	7	59
1	89	4	61	7	60
1	109	4	72	7	65
1	110	4	74	7	68
1	113	4	78	7	69
1	114	4	111	7	70
1	115	5	25	7	96
1	116	5	26	7	97
1	118	5	27	7	99
2	2	5	28	8	62
2	80	5	29	8	63
2	82	5	30	8	64
2	83	5	31	8	66
2	84	5	32	8	67
2	85	5	33	8	71
2	86	5	34	8	73
2	87	5	35	8	75
2	88	5	36	8	76
2	90	5	37	8	77
2	91	5	38	8	79
2	98	5	112		
3	7	6	39		
3	8	6	48		
3	9	6	92		
3	11	6	93		
3	12	6	94		
3	13	6	95		
3	14	6	100		
3	15	6	101		
3	16	6	102		
3	17	6	103		
3	18	6	104		
3	19	6	105		
3	20	6	106		
3	21	6	107		
3	22	6	108		
3	23				
3	117				

Tabla 3.5: Plantilla obtenida a partir de la agrupación PAM de 8 clústers. La plantilla se ordena de forma ascendente por el Id clúster. Los colores asignados a cada clúster se usan en las gráficas.

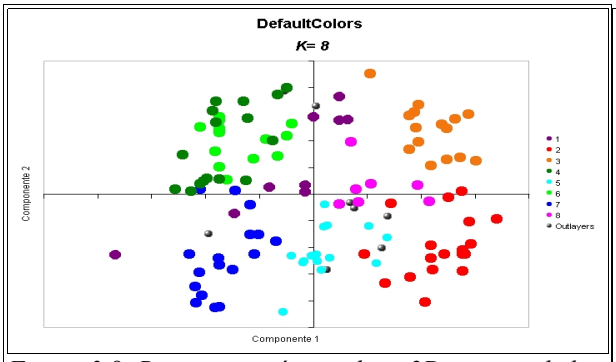


Figura 3.8: Representación en plano 2D a partir de las dos componentes principales obtenidas por el PC de la microarray m17.Los clústers se definen usando la plantilla default. Cada muestra se sitúa en el plano a partir de las coordenadas x e y (componente 1 y componente 2) y se colorea según el clúster al que pertenezca. El clúster Outlayers contiene condiciones muestrales que no han sido asignadas a ningún clúster, son condiciones muestrales outlayers

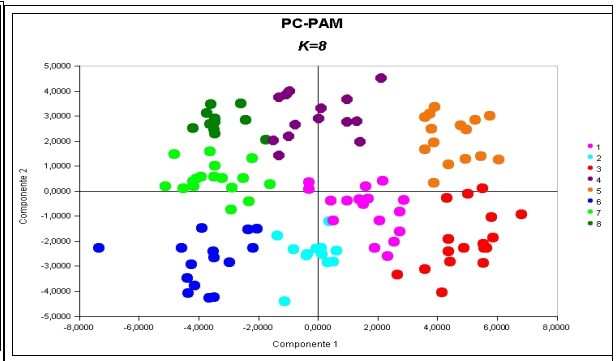


Figura 3.9: condiciones muestrales agrupadas en 8 clústers a partir de aplicar el algoritmo PAM a las componentes principales de la microarray m17. El eje x corresponde a la componente1 y el eje y al componente2. Cada muestra se colorea con el color del clúster al que pertenece. No hay outlayers

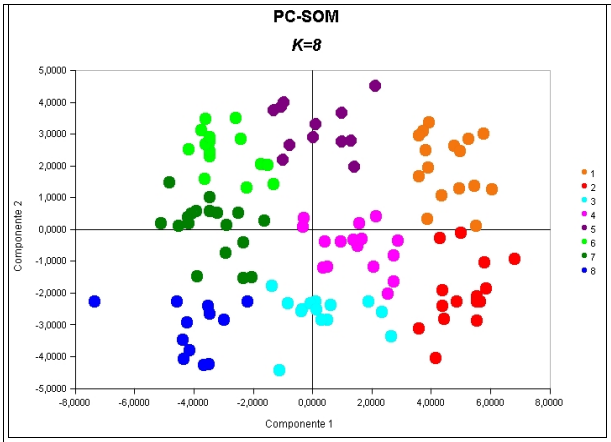


Figura 3.10: condiciones muestrales agrupadas en 8 clústers a partir de aplicar el algoritmo SOM a las componentes principales de la microarray m17. El eje x corresponde a la componente1 y el eje y al componente2. Cada muestra se colorea con el color del clúster al que pertenece. No hay outlayers

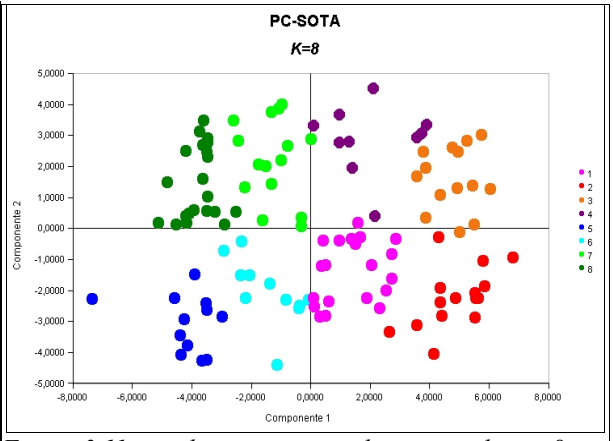


Figura 3.11: condiciones muestrales agrupadas en 8 clústers a partir de aplicar el algoritmo SOTA a las componentes principales de la microarray m17. El eje x corresponde a la componente1 y el eje y al componente2. Cada muestra se colorea con el color del clúster al que pertenece. No hay outlayers

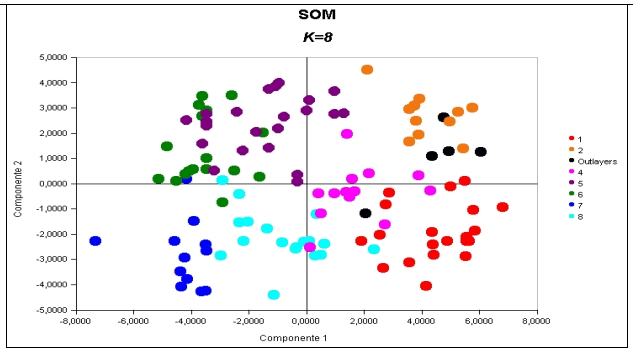


Figura 3.12: condiciones muestrales agrupadas en 8 clústers a partir de aplicar el algoritmo SOM directamente a la microarray m17. Para representar los datos se usaron las componentes principales calculadas a partir de la micorarray m17. El eje x corresponde a la componente1 y el eje y al componente2. Cada muestra se colorea con el color del clúster al que pertenece. Las condiciones muestrales de color negro son outlayers.

3.4.5.- Biclustering

Debida a la alta complejidad a la hora de interpretar y extraer los resultados obtenidos a partir del algoritmo de *biclustering* se opta por realizar solo el cálculo y guardar los resultados por si en un futuro se quiere realizar algún estudio sobre estos datos.

3.5.- Tratamiento de las condiciones muestrales outlayers

En los estudios anteriores no se controlaba la aparición de condiciones muestrales denominadas *outlayers*. La aparición de este tipo de condiciones muestrales puede llegar a dificultar la identificación de los posibles estados celulares que existan entre los diferentes clústers(cada clúster puede corresponder a un estado celular distinto). Por este motivo se intenta evitar en la medida de lo posible este tipo de condiciones muestrales.

La idea básica para solucionar este problema consiste en reasignar las condiciones muestrales *outlayers* a los clústers a los que pertenecía anteriormente (en la agrupación de *k-1* grupos) siempre y cuando se pueda. A este proceso le llamamos fusión de clústers.

En una primera versión para un conjunto de condiciones muestrales outlayers agrupadas en un mismo clúster y pertenecientes a una distribución de *k* clústers, se buscaba el clúster con mayor número de estas condiciones muestrales en la distribución de *k-1* clústers. A las condiciones muestrales pertenecientes a este clúster se le fusionaban los outlayers en la distribución de *k* clústers.

Esta solución no era correcta puesto que se estaban haciendo reasignaciones erróneas tal y como se muestra en la *fig. 3.13*.

Para solucionar el error se evalúan todos los clústers de la distribución de *k-1* clústers a los que pertenece cada uno de los outlayers realizando un proceso más exhaustivo.

La fusión correcta sería la que se muestra en la *fig. 3.14 .c*. Si buscamos los clústers a los que pertenecen los outlayers para la agrupación de k=2 obtenemos el clúster rojo y clúster verde. Para cada uno de estos dos clústers seleccionamos sus condiciones muestrales (para la agrupación de k=2) y buscamos el clúster al que pertenecen para el clusterng de k=3. Dos de estas condiciones muestrales se reasignarán al clúster rojo y las otras 3 al clúster verde.

De este manera hemos eliminado los *outlayers* reasignándolos a los clústers a los que pertenecían antes de realizar la agrupación de *k* grupos.

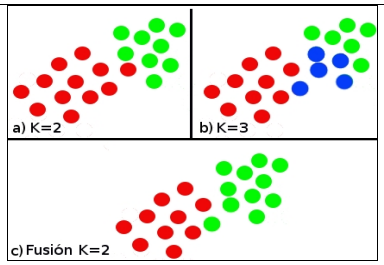


Figura 3.13:En la imagen b) se observan de color azul 5 outlayers. Para reasignarlos se usa la agrupación de k-1 (k=2) de la figura a). La fusión de la imagen c) corresponde a una primera aproximación del algoritmo. Se observa como hay dos condiciones muestrales mal reasignadas al clúster verde, pertenecen al clúster rojo.

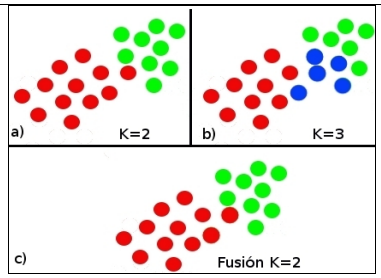


Figura 3.14: Resultados del algoritmo de fusión mejorado. Se observa como cada outlayer se a reasignado (imagen c) correctamente al clúster que le corresponde.

La fusión se realiza para todas las *k* de todos los algoritmos de agrupación implementados siempre y cuando sea posible.

Por especificación las plantillas deben contener identificadores clústers correlativos y sin saltos.

Por este motivo tanto para el fusionado como para el filtrado de *outlayers* es necesario una normalización de la plantilla que consiste en la reasignación de identificadores de los clústers.

Se guardan dos tipos de plantillas colors obtenidas para cada una de las *k's* de los distintos algoritmos de agrupación:

- Plantillas colors obtenidas sin realizar fusión y eliminando las condiciones muestrales *outlayers*
- Plantillas colors con fusión y eliminación de condiciones muestrales *outlayers*.

Los cálculos de integridad siempre se realizan después del fusionado y de la normalización ya que tanto el número de clústers (*k*) como las condiciones muestrales que contiene cada clústers pueden variar antes y después del fusionado.

3.6.- Integración de la agrupación en el preproceso

Llamamos preproceso al proceso principal que se realiza automáticamente al cargar una microarray al sistema (servidor). Este preproceso solo se realiza una vez por microarray.

El preproceso está compuesto por varios subprocesos. Uno de estos subprocesos corresponde a los diferentes métodos de agrupación implementados.

Este subproceso a parte de realizar los cálculos de agrupación es el encargado de guardar todos los resultados en ficheros generando así una estructura de ficheros que se describe en la *sección 4.10.2.*

3.7.- Gestión de resultados de la agrupación

Para gestionar tanto los resultados obtenidos por la aplicación desarrollada como para gestionar un gestor de plantillas de usuarios *(histórico de plantillas)* se pensó inicialmente en usar una base de datos.

Después de analizar el problema se llegó a la conclusión de que no era necesaria una base de datos para gestionar estos datos ya que el volumen de información a guardar en la base de datos era muy pequeña.

Se optó por una simple gestión de ficheros que era más rápida de implementar.

3.8.- Interfaz web: Integrar los resultados de la agrupación en el aplicativo web

Una vez se implementaron todos los algoritmos y se validaron se procedió a integrar los resultados obtenidos en la aplicación web (applet) PCOPGene[14].

Antes de describir como se integraron los resultados a la aplicación se hará una breve explicación de la funcionalidad de la aplicación de la parte que nos atañe.

Es necesario que el usuario se registre como nuevo usuario en el servidor[38] o que acceda en modo invitado. Para poder realizar cualquier tipo de operación los usuarios deben tener por lo menos una microarray asignada.

Cada usuario tiene un ficheros de clústers propio.

Es importante destacar esta plantilla porque es la que se usa para todas las operaciones relacionadas con la agrupación de condiciones muestrales de las aplicaciones web del servidor. Se la conoce como plantilla actual.

Esta plantilla es la que se modifica o actualiza con otras agrupaciones de condiciones muestrales que el usuario decida.

Para acceder a esta funcionalidad se dispone de un botón en la pantalla principal de la aplicación con el texto "*Update condiciones muestrales-Class Palette*" *(fig. 3.14).*

Este botón abre automáticamente una página web desarrollada en PHP donde se le añaden una serie de funcionalidades a parte de las dos que ya tenía. En la *fig. 3.15* se muestra como estaba la página antes de la modificación.

A través de esta página web se podían realizar dos operaciones:

- Cargar un fichero de agrupación de condiciones muestrales externo (paleta externa).
- Definir manualmente los clústers, es decir, asignando manualmente cada muestra al clúster que el usuario requiera.

Las dos funcionalidades principales que se añaden son:

- Listado de los ficheros de clústers precalculados
- Permitir al usuario actualizar la paleta de colores actual con uno de los ficheros de clústers precalculados en el preproceso de la microarray y permitir la descarga de éstas.

– Crear un histórico de plantillas de colores del usuario permitiendo actualizar la paleta actual con alguna de estas paletas guardadas ,permitir eliminarlas y permitir también su descarga.

La fig. *3.16* muestra el resultado después de la modificación.

Después de implementar esta funcionalidad se encontraron una serie de deficiencias funcionales a nivel de usuario.

Uno de estos problemas era que el listado de ficheros era muy extensos. Analizando los resultados se encontró que había plantillas de colores de un mismo método de agrupación que contenían una agrupación de condiciones muestrales idéntica.

Para solventarlo se implementa un filtro para detectar estos casos.

Con este proceso se eliminaron bastantes ficheros de plantillas disminuyendo considerablemente la lista de plantillas.

Otro de los problemas encontrados es que dado que se guardaban las mejores plantillas de cada algoritmo según el índice de integridad Dunn y el índice Silhouette, de forma visual era difícil identificar cual era el índice que había marcado esa plantilla como la mejor de todas. Se solucionó simplemente resaltando el valor del índice de integridad para el cual esa plantilla era la mejor entre todas (*fig 3.16*).

Finalmente para orientar mejor al usuario a la hora de analizar las plantillas obtenidas en el preproceso se optó por añadir dos nuevas funcionalidades, una relacionada con la manera de mostrar este listado y la otra con una normalización interna de todas las plantillas.

En vez de listar por orden alfabético se decidió realizar una ordenación previa de las plantillas precalculadas obtenida a partir del *hierarchical clustering* de estas.

El proceso de normalización se realiza a partir de esta agrupación obtenida.

Tanto la ordenación como la normalización se detallan en la siguiente sección.

Para añadir el histórico de plantillas de colores de usuario se añadió un botón que permite guardar el fichero de clústers actual con el nombre de plantilla que el usuario quiera (*fig. 3.17*).

A este histórico se le añaden las siguiente funcionalidades:

– Eliminar y descargar plantillas del histórico del usuario (*fig. 3.17*)

– Actualizar la paleta de colores actual con una de las plantillas del histórico del usuario (*fig. 3.17*).

Tal y como se hacía con los ficheros de clústers precalculados se listan las plantillas propias del usuario(*fig. 3.17*).

Al igual que el listado de plantillas de colores precalculadas se añadió una funcionalidad que facilita al usuario el análisis de sus plantillas de colores guardadas. Mediante un botón se le permite al usuario normalizar todos los ficheros del histórico(*fig. 3.17*). Este proceso es el mismo que se aplica en los ficheros de plantillas de colores precalculados y se describe a continuación.

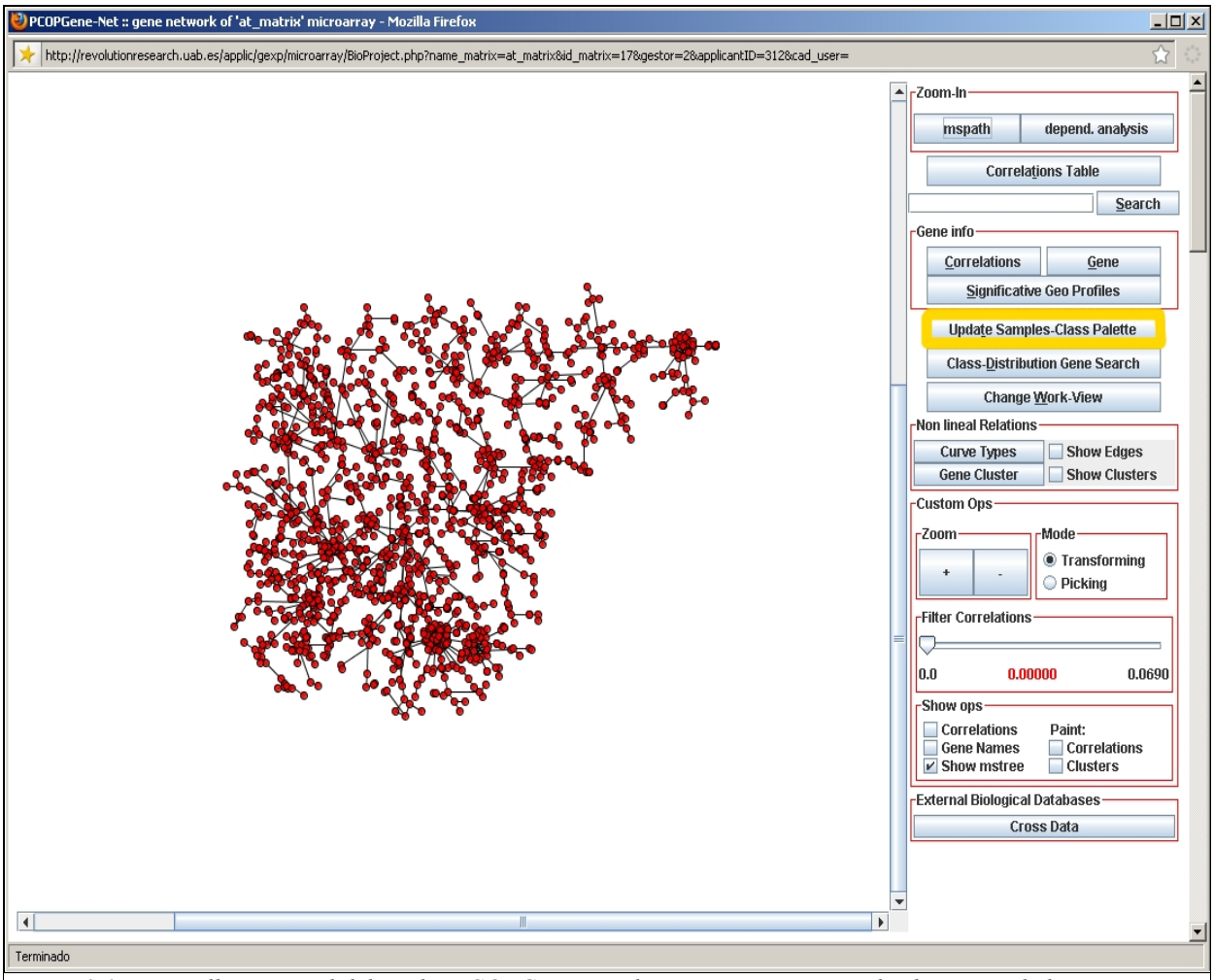


Figura 3.14: Pantalla principal del applet PCOPGene. En el centro se muestran todos los genes de la microarray correlacionados. A la derecha todas las operaciones posibles sobre esta microarray (m17). Se remarca en amarillo la funcionalidad que permite modificar el fichero de clústers actual.

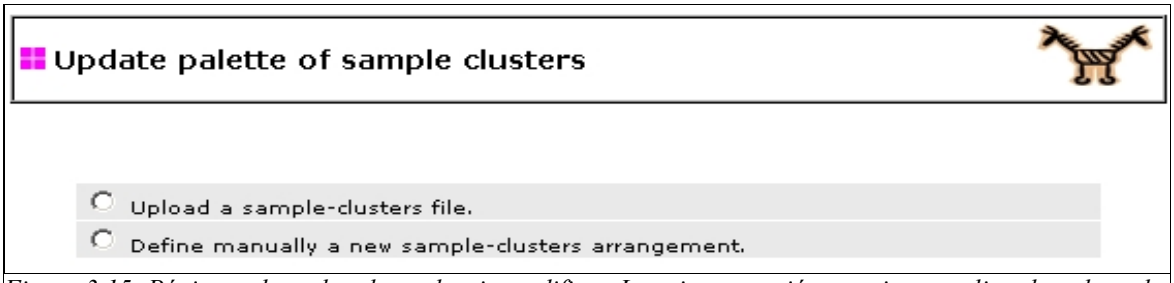
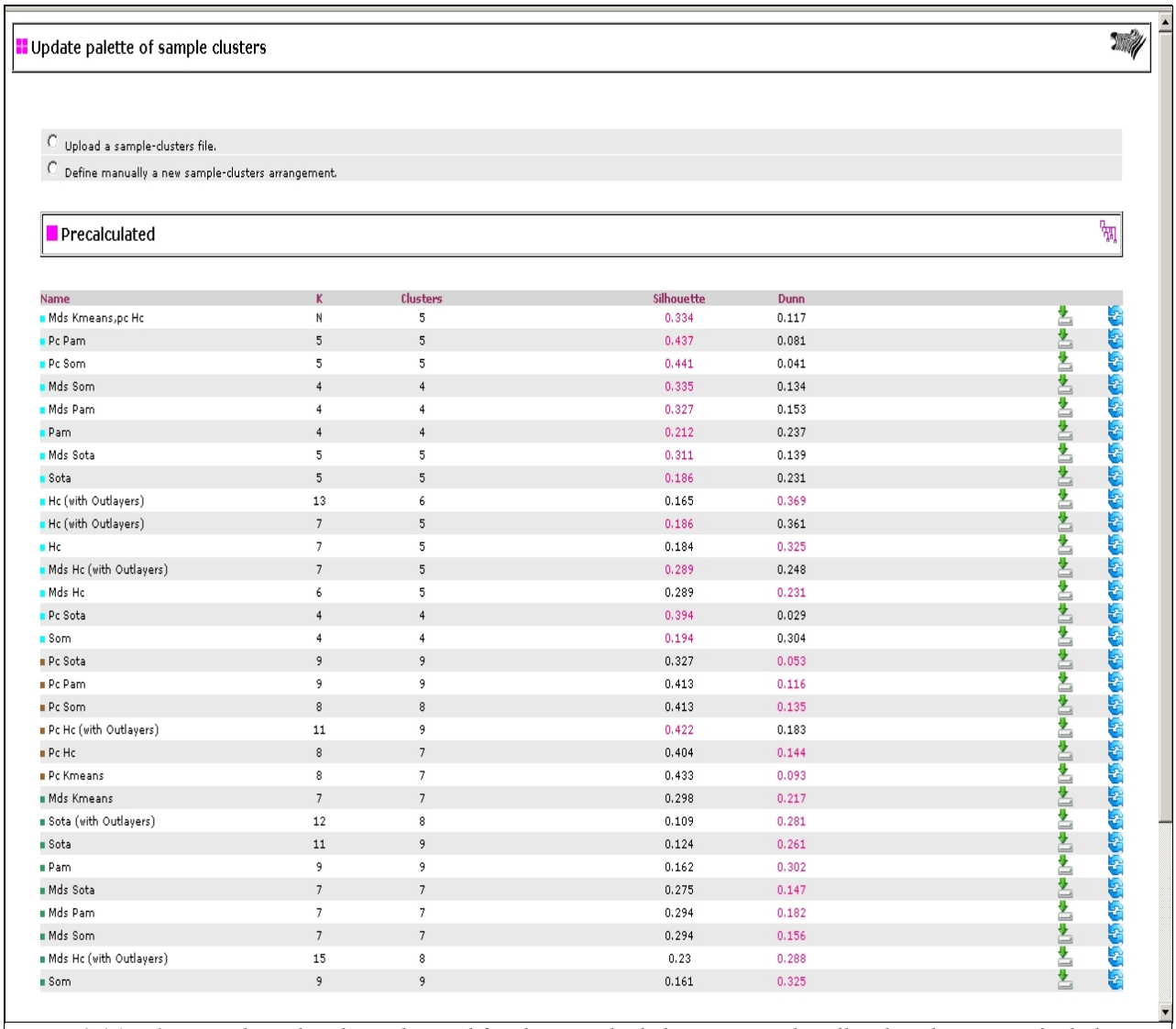


Figura 3.15: Página web appl_palette.php sin modificar. La primera opción permite actualizar la paleta de colores actual con un fichero externo. La segunda opción permite al usuario asignar manualmente cada una de las condiciones muestrales al clúster que le convenga



Name	K	Clusters	Silhouette	Dunn
Mds Kmeans,pc Hc	N	5	0.334	0.117
Pc Pam	5	5	0.437	0.081
Pc Som	5	5	0.441	0.041
Mds Som	4	4	0.335	0.134
Mds Pam	4	4	0.327	0.153
Pam	4	4	0.212	0.237
Mds Sota	5	5	0.311	0.139
Sota	5	5	0.186	0.231
Hc (with Outlayers)	13	6	0.165	0.369
Hc (with Outlayers)	7	5	0.186	0.361
Hc	7	5	0.184	0.325
Mds Hc (with Outlayers)	7	5	0.289	0.248
Mds Hc	6	5	0.289	0.231
Pc Sota	4	4	0.394	0.029
Som	4	4	0.194	0.304
Pc Sota	9	9	0.327	0.053
Pc Pam	9	9	0.413	0.116
Pc Som	8	8	0.413	0.135
Pc Hc (with Outlayers)	11	9	0.422	0.183
Pc Hc	8	7	0.404	0.144
Pc Kmeans	8	7	0.433	0.093
Mds Kmeans	7	7	0.298	0.217
Sota (with Outlayers)	12	8	0.109	0.281
Sota	11	9	0.124	0.261
Pam	9	9	0.162	0.302
Mds Sota	7	7	0.275	0.147
Mds Pam	7	7	0.294	0.182
Mds Som	7	7	0.294	0.156
Mds Hc (with Outlayers)	15	8	0.23	0.288
Som	9	9	0.161	0.325

Figura 3.16: Página web appl_palette.php modificada. Listado de las mejores plantillas de colores precalculadas en el preproceso de la microarray 17. Se muestran 5 campos de información: nombre del algoritmo de agrupación, número de clústers inicial, número de clústers final después de aplicar posibles fusiones o eliminaciones de condiciones muestrales, valor del índice de integridad Dunn y Silhouett. Se resalta con color violeta el valor del índice de integridad que hace de esa plantilla sea la mejor para su algoritmo de agrupación. En las dos columnas finales se añaden dos botones para cada plantilla , el primero permite la descarga de dicha plantilla y el segundo actualiza el fichero de clústers actual por la seleccionada.

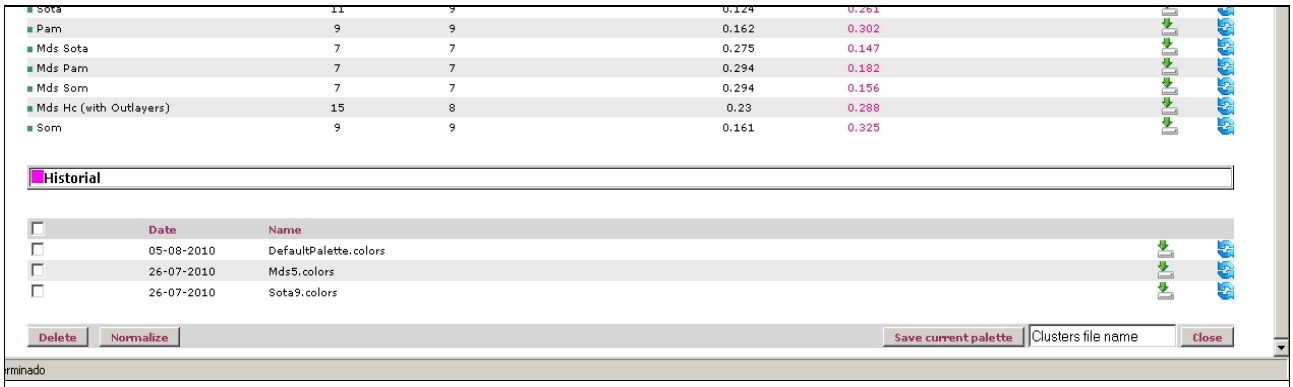


Figura 3.17: Página web appl_palette.php modificada. Se listan los ficheros del usuario ubicados en el directorio Colors Historic de la microarray. Se añade la opción de eliminado de plantillas del histórico y guardado de la plantilla actual con el nombre designado por el usuario. A parte en las dos últimas columnas se añaden dos botones, uno para descargar la plantilla descargada y el otro para actualizar la plantilla actual con la plantilla seleccionada. El botón Normalize ordena y normaliza los ficheros del histórico del usuario basado en el hierarchical clustering de estos ficheros

3.9.- Ordenación, agrupación y normalización de los ficheros de clústers

Es normal que los diferentes métodos de agrupación, para los mismos datos, generen agrupaciones muy similares o idénticas. Sería muy útil para el usuario poder agrupar las plantillas por similitud y poder identificar las plantillas que sean equivalentes.

Por ello se utiliza la agrupación basada en el algoritmo Hierarchical que a parte de ordenar las plantillas por disimilitud (distancia) permite agruparlas por similitud.

A partir de estos grupos obtenidos se realiza una normalización de cada uno de los ficheros de clústers de cada grupo que consiste en buscar un fichero *guía* para cada grupo y a partir de este reasignar los identificadores de los clústers internos de cada fichero que no sea el *guía* y que pertenezca a su grupo.

Estas mejoras se implementaron como procesos independientes de la agrupación, de este modo se podrían aplicar a cualquier conjunto de plantillas de colores.

La agrupación de las plantillas se realiza a partir de las distancias mínimas calculadas entre cada plantilla.

Para el cálculo de la distancias entre dos plantillas se hizo una primera aproximación que consistía en calcular de forma exhaustiva las distancias de todas las combinaciones posibles de pares de clústers de ambas plantillas. De todas las combinaciones se escogen las de menor distancia.

Este proceso es muy costoso a nivel de tiempo así que se optó escoger de las dos plantillas la de menor número de clústers y a partir de estos clústers calcular las distancias con los clústers de la otra plantilla reduciendo así el número de combinaciones posibles.

En el ejemplo de la tabla *3.6* muestra dos plantillas de colores que representan unas agrupaciones de condiciones muestrales. La plantilla A agrupa 15 condiciones muestrales en 3 clústers, en cambio , la plantilla B agrupa las 15 condiciones muestrales en 4 clústers.

Para cada clúster de la plantilla A se calcula la distancia con cada uno de los clústers de la plantilla B (*tabla 3.7*). Para cada clúster de la plantilla A se escoge la pareja de clústers que menor distancia tenga. Estas parejas (asignaciones) se usarán más tarde para el proceso de normalizado.

La distancia total entre las dos plantillas es la suma de estas distancias mínimas, es decir, 4 en el caso del ejemplo.

Un problema que se encuentra es que hay casos en los que ciertos clústers de la plantilla A tienen el mismo clúster candidato (clúster más cercano) de la plantilla B.

En la *tabla 3.8* se muestra una matriz de distancias entre dos agrupaciones , una de 5 clústers y la otra de 6. Buscando la distancia mínima total se ha encontrado que el clúster 2 y 3 tienen como clúster más

cercano de la plantilla B el 3.

Para evitar estos casos lo que se hace es calcular de forma exhaustiva las distancias mínimas usando solo las permutaciones entre estos tipos de clústers y los clústers repetidos de la plantilla B más los clústers sin asignar de esta misma plantilla.

En la tabla *3.9* se calcula la distancia de cada una de las permutaciones entre los clústers (2,3) de la plantilla A y (2,3,6) de la plantilla B. En este último grupo se añaden todos los clústers de la plantilla B que no fueron seleccionados como candidatos (el clúster 6 en el ejemplo).

Realizadas estas mejoras se obtiene una distancia mínima cercana a la óptima ya que como se indicó antes buscar la distancia óptima de modo exhaustivo ralentizaba mucho el proceso.

Paleta colores A
1- 1,2,3,4,5
2- 6,7,8,9,10
3- 11,12,13,14,15
Paleta colores B
1- 1,2,3
2- 4,5,6,7
3- 11,12,13,14,15
4- 8,9,10

Tabla 3.6: Simulación de paletas de colores. La plantilla A tiene 15 condiciones muestrales agrupadas en 3 clústers. La plantilla B tiene 15 condiciones muestrales agrupadas en 4 clústers

	1	**2**	**3**	**4**
1	**2**	5	10	8
2	7	5	10	**2**
3	8	9	**0**	8

Tabla 3.7: Distancias entre los clústers de la plantilla A y la plantilla B. Cada fila corresponde a un clúster de la plantilla A y las columnas a cada clúster de la plantilla B. Por ejemplo, entre el clúster 1 de la plantilla A y el clúster 2 de la plantilla B hay 5 condiciones muestrales de diferencia, las condiciones muestrales 1,2,3 del clúster 1 de la plantilla A no están en el clúster 2 de la plantilla B y a la inversa las condiciones muestrales 6 y 7 están en la plantilla B pero no en la plantilla A. En rojo se resaltan las distancias mínimas, es decir las parejas de clústers más cercanas. La suma de estas corresponde a la distancia entre ambas plantillas de colores, 4

	1	**2**	**3**	**4**	**5**	**6**
1	**15**	23	43	43	63	67
2	25	17	**11**	15	35	39
3	37	29	**9**	10	43	47
4	36	28	22	**18**	34	32
5	74	66	60	60	**30**	32

Tabla 3.8: Distancias entre dos plantillas de colores, una de 5 clústers (plantilla A) y la otra de 6(plantilla B).Buscamos los clústers de la plantilla B más cercanos a cada clúster de la plantilla A (marcados en rojo). El clúster más cercano para los clusters 2 y 3 de la plantilla A es el mismo, 3. Se tiene que buscar otra combinación para los clústers 2 y 3 de la plantilla A tal que la suma de las distancias de los dos clúster respecto los clústers candidatos de la plantilla B sea la mínima y estos clústers sean diferentes

		DISTANCIA
2-2	**3-3**	**26**
2-2	3-6	56
2-3	3-2	40
2-3	3-6	58
2-6	3-2	68
2-6	3-3	48

Tabla 3.9: Distancias de las permutaciones sin repetición entre los clústers (2,3) de la plantilla A y los clústers (2,3 ,6) de la plantilla B. La combinación con menor distancia es la que se remarca en rojo (26)..

Finalmente se encontró un error en el computo total de la distancia.

No se estaba penalizando el hecho de que dos plantillas tuvieran número diferente de clústers.

Para solventarlo a la distancia total encontrada se le suman las condiciones muestrales de la plantilla con mayor numero de clústers que pertenezcan a un clúster sin asignar.

En la tabla *3.8* el clúster 6 se queda sin asignación respecto a los clústers de la plantilla A, sus condiciones muestrales se sumarían a la distancia total encontrada.

La distancia total se usa para construir la matriz de distancias entre todas las plantillas del directorio tratado y para identificar en un proceso posterior plantillas equivalentes (con agrupaciones idénticas). Las parejas de plantillas que tengan una distancia total de 0 quiere decir que son plantillas con agrupaciones idénticas.

El proceso de normalización se añadió con tal de facilitar el análisis de las plantillas que contiene cada clúster encontrado.

Este proceso consiste en buscar una plantilla guía para cada grupo de plantillas.

Para cada grupo de plantillas se modifica el identificador de los clústers de las plantillas teniendo en cuenta los identificadores de los clústers de la plantilla guía de dicho grupo.

Siguiendo el ejemplo de la tabla *3.6* supongamos que ambas plantillas pertenecen al mismo grupo de plantillas y que la plantilla A es la plantilla *guía.*

A partir de la matriz de distancias encontrada en la tabla *3.7* encontramos las mejores asignaciones correspondientes a las distancias mínimas entre los clústers de estas dos plantillas(*tabla 3.10*).

Analizando estas asignaciones se observa :

- Los clústers 1 y 3 de la plantilla de colores B se mantienen igual ,
- Al clúster 4 se le reasigna un nuevo identificador, el 2 que corresponde al identificador del clúster más cercano en la plantilla A.
- Dado que las paletas tiene identificadores correlativos y sin repetición se han de reasignar el resto de identificadores de la plantilla B para que se cumpla esta condición. En este caso hay que reasignar el identificador del clúster 2 que no tiene asignación. El identificador nuevo es el siguiente al identificador mayor asignado, el mayor asignado es 3 , por lo tanto al clúster 2 de la plantilla B le corresponde el nuevo identificador 4.

En las figuras *3.18 ,3.19* y *3.20* tenemos una representación gráfica de un conjunto de condiciones muestrales cualquiera correspondientes a dos genes cualquieras X e Y. Las *fig. 3.18 y 3.19* se agrupan siguiendo las agrupaciones de las plantillas A y B . En la *fig. 3.20* las condiciones muestrales se agrupan siguiendo la agrupación de la plantilla B pero con los identificadores de los clústers reasignados.

Analizando las *fig. 3.18* y *3.19* se hace difícil encontrar que subdivisión de las condiciones muestrales se ha llevado a cavo. En cambio, si comparamos las figuras *3.18 y 3.20* se ve fácilmente como el clúster 4 es la nueva subdivisión y como el resto de clústers no varían.

Id Clúster B	IdClúster A	Distancia
1	1	2
2	-	-
3	3	0
4	2	2

Tabla 3.11: A partir de la matriz de distancias entre las plantillas A y B (tabla 3.8) encontramos las mejores asignaciones correspondientes a las distancias mínimas. De este modo los identificadores de clústers 1 y 3 de la plantilla B se mantienen y se reasigna el identificador 4 con el identificador 2. El identificador 2 de la plantilla B no tiene asignación por lo tanto se le asigna el identificador correlativo al mayor de los asignados, es decir, el mayor asignado es 3 por lo tanto al clúster 2 de la plantilla B se le asigna el identificador 4.

Paleta colores A
1- 1,2,3,4,5
2- 6,7,8,9,10
3- 11,12,13,14,15
Paleta colores B
1- 1,2,3
4- 4,5,6,7
3- 11,12,13,14,15
2- 8,9,10

Tabla 3.12: Resultado de la normalización de la plantilla B respecto a su paleta quía, la paleta A

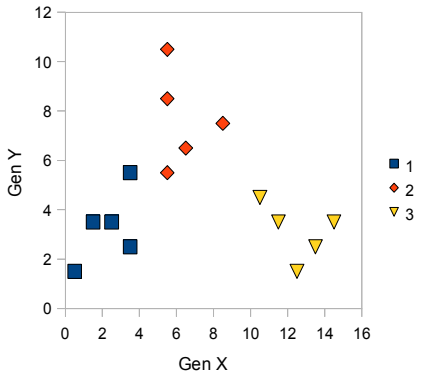


Figura 3.18: A partir de 15 condiciones muestrales creadas para dos genes X e Y se agrupan en 3 clústers a partir de la plantilla de colores A de la tabla 3.7.

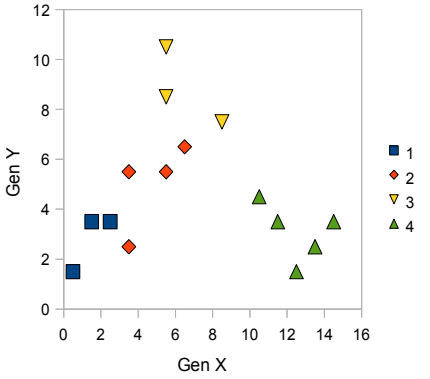


Figura 3.19: A partir de 15 condiciones muestrales creadas para dos genes X e Y se agrupan en 3 clústers a partir de la plantilla de colores B de la tabla 3.7. Analizando visualmente cuesta encontrar como se ha pasado de 3 a 4 clústers. Por ello se normaliza la plantilla.

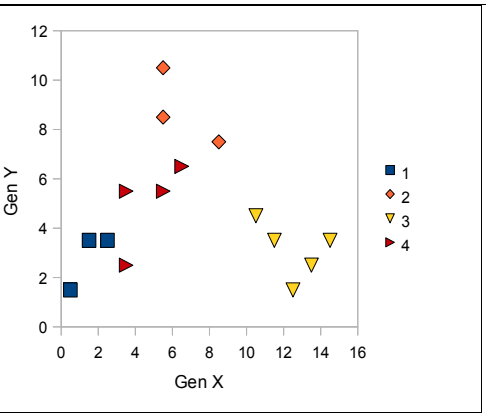


Figura 3.20:A partir de 15 condiciones muestrales creadas para dos genes X e Y se agrupan en 3 clústers a partir de la plantilla de colores B de la tabla 3.12 que corresponde a la plantilla con los identificadores de clústers reasignados. Con esta normalización vemos claramente como el clúster 4 es la nueva división que se obtuvo a partir de la agrupación de la plantilla B.

A través del botón de normalizado *(fig. 3.17)* el usuario puede normalizar sus plantillas.

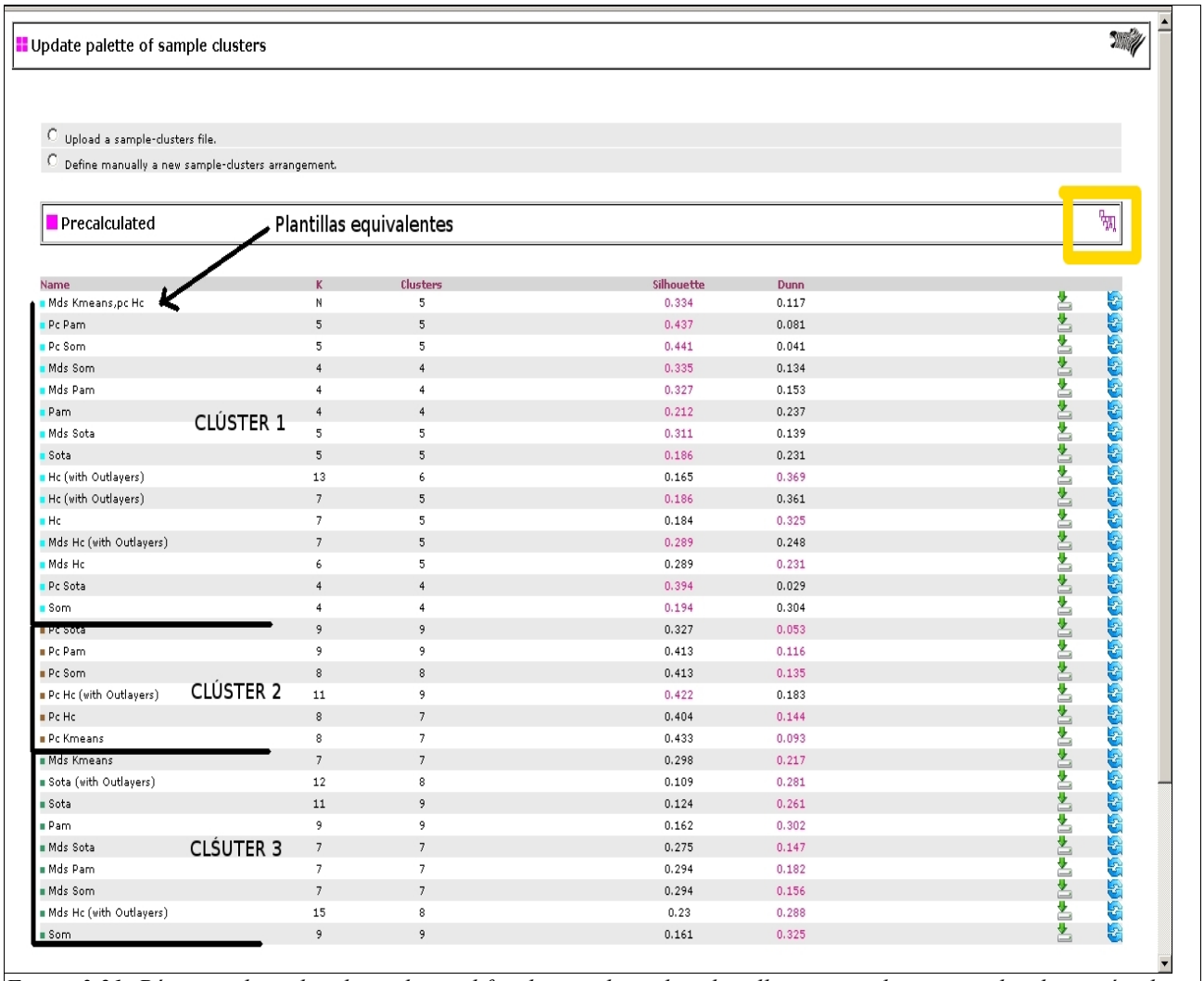
Ahora en vez de listar directamente las plantillas de colores precalculadas se muestran agrupando las plantillas por clústers e identificando las plantillas que son equivalentes tal y como se muestra en la *fig. 3.21*.Cada clúster de plantillas se identifica con un color propio.

Para ver un resumen de la agrupación realizada sobre las plantillas precalculadas se habilita un icono (*fig.3.21*) que redirecciona al usuario a una nueva página web implementada (*fig.3.22*)

En la esquina superior derecha se muestra el dendrograma resultante. Mediante este dendrograma se puede observar como se han agrupado las plantillas, cuanto más alejadas estén las plantillas en el gráfico mayor distancia hay entre ellas.

Por último, se muestra la tabla de distancias entre cada una de las plantillas de colores del directorio tratado. Es una matriz triangular superior[i]. Esta es la matriz a partir de la cual se realizó el *hierarchical clustering*.

i) *Una matriz triangular superior es un tipo especial de matriz cuadrada cuyos elementos por encima de su diagonal principal son cero*



Name	K	Clusters	Silhouette	Dunn
Mds Kmeans,pc Hc	N	5	0.334	0.117
Pc Pam	5	5	0.437	0.081
Pc Som	5	5	0.441	0.041
Mds Som	4	4	0.335	0.134
Mds Pam	4	4	0.327	0.153
Pam	4	4	0.212	0.237
Mds Sota	5	5	0.311	0.139
Sota	5	5	0.186	0.231
Hc (with Outlayers)	13	6	0.165	0.369
Hc (with Outlayers)	7	5	0.186	0.361
Hc	7	5	0.184	0.325
Mds Hc (with Outlayers)	7	5	0.289	0.248
Mds Hc	6	5	0.289	0.231
Pc Sota	4	4	0.394	0.029
Som	4	4	0.194	0.304
Pc Sota	9	9	0.327	0.053
Pc Pam	9	9	0.413	0.116
Pc Som	8	8	0.413	0.135
Pc Hc (with Outlayers)	11	9	0.422	0.183
Pc Hc	8	7	0.404	0.144
Pc Kmeans	8	7	0.433	0.093
Mds Kmeans	7	7	0.298	0.217
Sota (with Outlayers)	12	8	0.109	0.281
Sota	11	9	0.124	0.261
Pam	9	9	0.162	0.302
Mds Sota	7	7	0.275	0.147
Mds Pam	7	7	0.294	0.182
Mds Som	7	7	0.294	0.156
Mds Hc (with Outlayers)	15	8	0.23	0.288
Som	9	9	0.161	0.325

Figura 3.21: Página web appl_palette.php modificada para listar las plantillas agrupadas y normalizadas según el algoritmo hierarchical clustering. A cada clúster de plantillas se le asigna un color, en este caso existen 3 clústers de plantillas. En la primera fila se puede identificar dos plantillas que son equivalentes, es decir, contienen una misma agrupación. Para poder ver un resumen de los resultados se habilita un icono con forma de dendrograma (remarcado con cuadro amarillo). Accionando este icono podemos acceder a otra página web (appl_palette_dendogram.php) donde se muestran una serie de resultados.

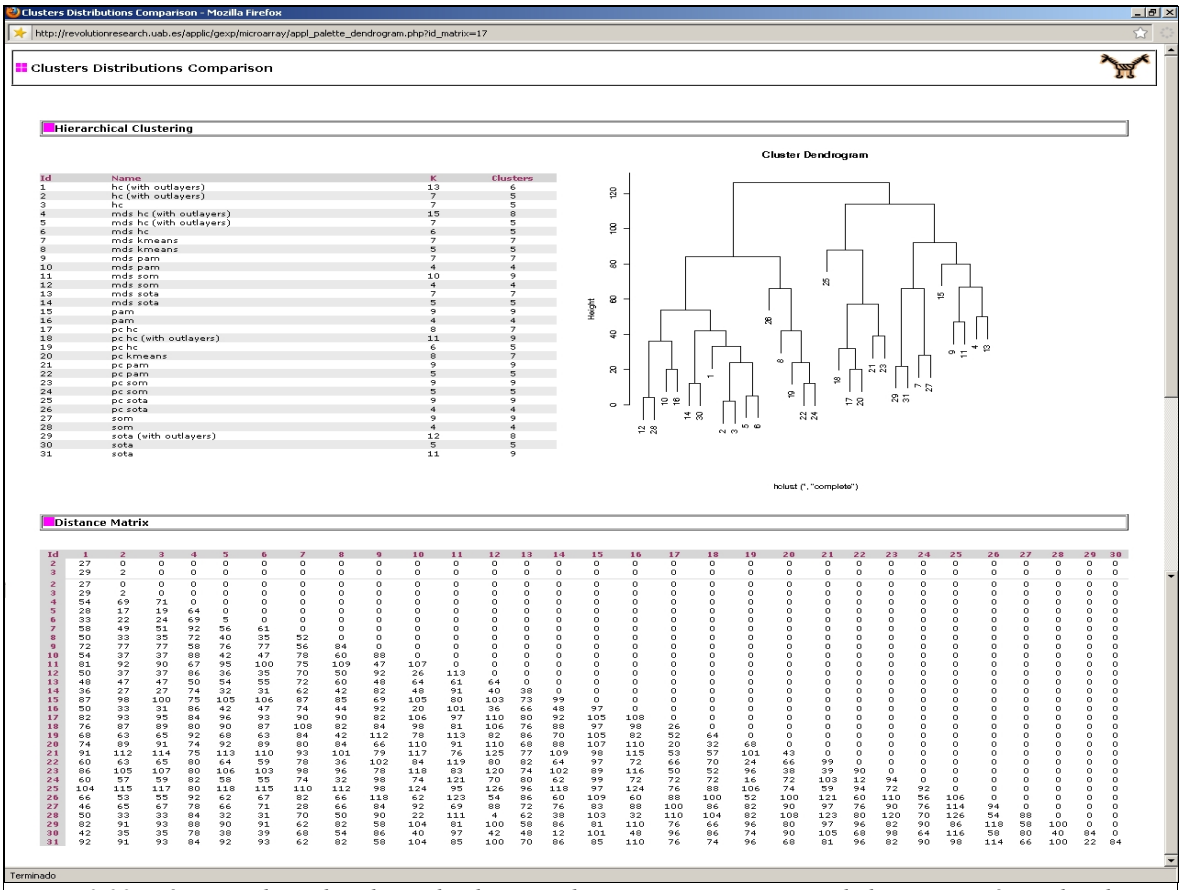


Figura 3.22: Página web appl_palette_dendogram.php. Muestra un resumen de la agrupación realizada sobre las plantillas precalculadas para una microarray. Se muestra una tabla con 4 campos de información: identificado asignado a cada una de las plantillas, nombre de la plantilla , K o número de clústers inicial y número de clústers final (después de realizar proceso de fusión o descarte). A la derecha se muestra el dendrograma resultante de realizar el algoritmo jerárquico completo sobre estas plantillas, agrupa las plantillas separando las plantillas más distanciadas entre sí. Finalmente se muestra una matriz de distancias entre cada una de las plantillas.

3.10.- Búsqueda de genes marcadores

Esta fase constituye una fase independiente a la primera.

Consideramos que los diferentes clústers de las condiciones muestrales representan diferentes estados celulares porque los genes se expresan de forma similar para unos clústers y de forma diferente para otros clústers.

Teniendo en cuenta esto, cuando buscamos un gen en que se sobre expresa en un determinado clúster respecto al resto de clústers obtenemos un gen que se expresa de manera diferente en dos estados celulares diferentes. Cuanto mayor distancia haya entre los dos clústers más le afecta al gen el cambio de estado celular.

Por este motivo es interesante ofrecer al usuario una herramienta que le permita realizar búsquedas de genes para los cuales los clústers de condiciones muestrales cumplan una serie de condiciones determinadas. Con ello el usuario puede llegar a encontrar genes marcadores.

Para la búsqueda de estos genes se utilizó como base matemática los intervalos de confianza creados a partir de la distribución T d Student (variación de la distribución normal estándar).

Definiendo los intervalos de confianza para cada uno de los clústers en los que se clasificaron las condiciones muestrales podemos validar las condiciones de búsqueda que deben cumplir para cada gen de la microarray.

Las condiciones que se pueden dar son las siguientes:

- Que para un clúster determinado un gen se sobre exprese sobre el valor basal 0, es decir , el conjunto de condiciones muestrales del clúster de este gen está por encima de 0.
- Que para un clúster determinado un gen se info exprese sobre el valor basal 0, es decir , el conjunto de condiciones muestrales del clúster de este gen está por debajo de 0.
- Que para un par de clústers un gen se sobre exprese en uno de ellos respecto al otro o a la inversa.
- Que para un par de clústers un gen se exprese de manera diferente, es decir, no haya solapamiento entre estos dos clústers.

El usuario es quien marca estas condiciones y además selecciona el nivel de confianza (amplitud de los intervalos).

Este nivel de confianza define la cantidad de condiciones muestrales que deben estar dentro del intervalo. Se fijan los siguientes: 99.7%, 99.1% y 95.1%. Cuanto mayor nivel de confianza se exija más restrictiva será la búsqueda. Para la distribución T d Student los intervalos que corresponden a estos niveles son los siguientes:

- 99.7% de las observaciones están entre los valores 2.968 y -2.968
- 99.1% de las observaciones están entre los valores 2.612 y -2.612
- 95.1% de las observaciones están entre los valores 1.969 y -1.969

En la gráfica de la *fig. 3.23* se muestra un ejemplo de una distribución normal de las condiciones muestrales de la microarray *m17* agrupadas en 5 clústers para el gen PME-1. Cada par de lineas verticales del mismo color representan los intervalos de confianza para cada clúster con un 99.31% de probabilidades de que las condiciones muestrales de cada clúster se ubiquen en estos intervalos. A partir de estos intervalos podemos observar como por ejemplo el clúster 1 se info expresa respecto el basal porque su intervalo cae por debajo del valor basal 0, la distancia en este caso sería 0,17.

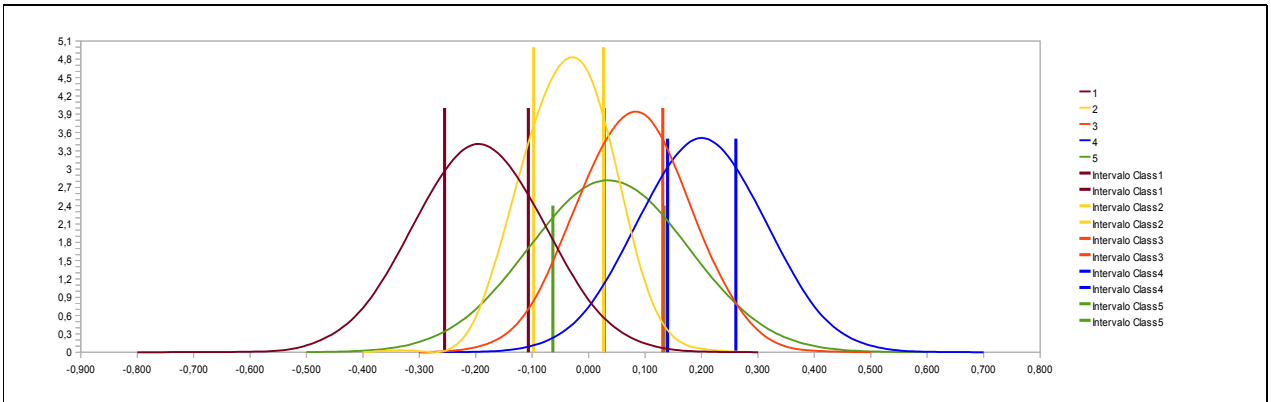


Figura 3.23: Distribución normal de las condiciones muestrales de la microarray 17 agrupadas en 5 clústers para el gen PME-1. Las lineas verticales del mismo color representan los intervalos de confianza para cada clúster con un 99.31% de probabilidades de que las condiciones muestrales de estos clústers se ubiquen en estos intervalos. Se puede ver como el clúster 1 se info expresa respecto el basal 0 con una distancia de 0,17 y los clústers 3 y 4 se sobre expresan respecto el basal con distancias 0.028 y 0.14 respectivamente. También se puede observar como el clúster 2 se sobre expresa respecto el clúster 1 pero con una distancia muy pequeña de 0.009. El 4 se sobre expresa respecto el resto de clústers.

3.11.- Interfaz web: Integrar la búsqueda de genes marcadores en el aplicativo web

La integración de esta nueva funcionalidad de búsqueda de genes a partir de los intervalos de confianza se agrega en una página web existente disponible a través del applet PCOPGene[14].

En esta página el usuario puede definir una serie de condiciones sobre cada clúster (los clústers corresponden a la agrupación de la plantilla actual del usuario) . Condiciones respecto el basal 0 y condiciones entre clústers.

A esta página se le añadió dos elementos nuevos:

– Una lista para que el usuario escoja el nivel de confianza (0.003, 0.009 y 0.049)

– Un botón para realizar la búsqueda usando los intervalos de confianza.

En la *fig.3.24* se muestra la página modificada.
La *fig. 3.25* muestra el resultado de una búsqueda.

El listado de genes encontrado se ordena por la distancia encontrada para cada gen de mayor a menor.

Cuanto mayor sea la distancia mejor calificación tiene el gen para ser un gen marcador. El degradado de color de la columna Dist muestra cuan similares son las distancias, cuanto más degradado mayor separación de distancias y como consecuencia mejor distinción del gen marcador.

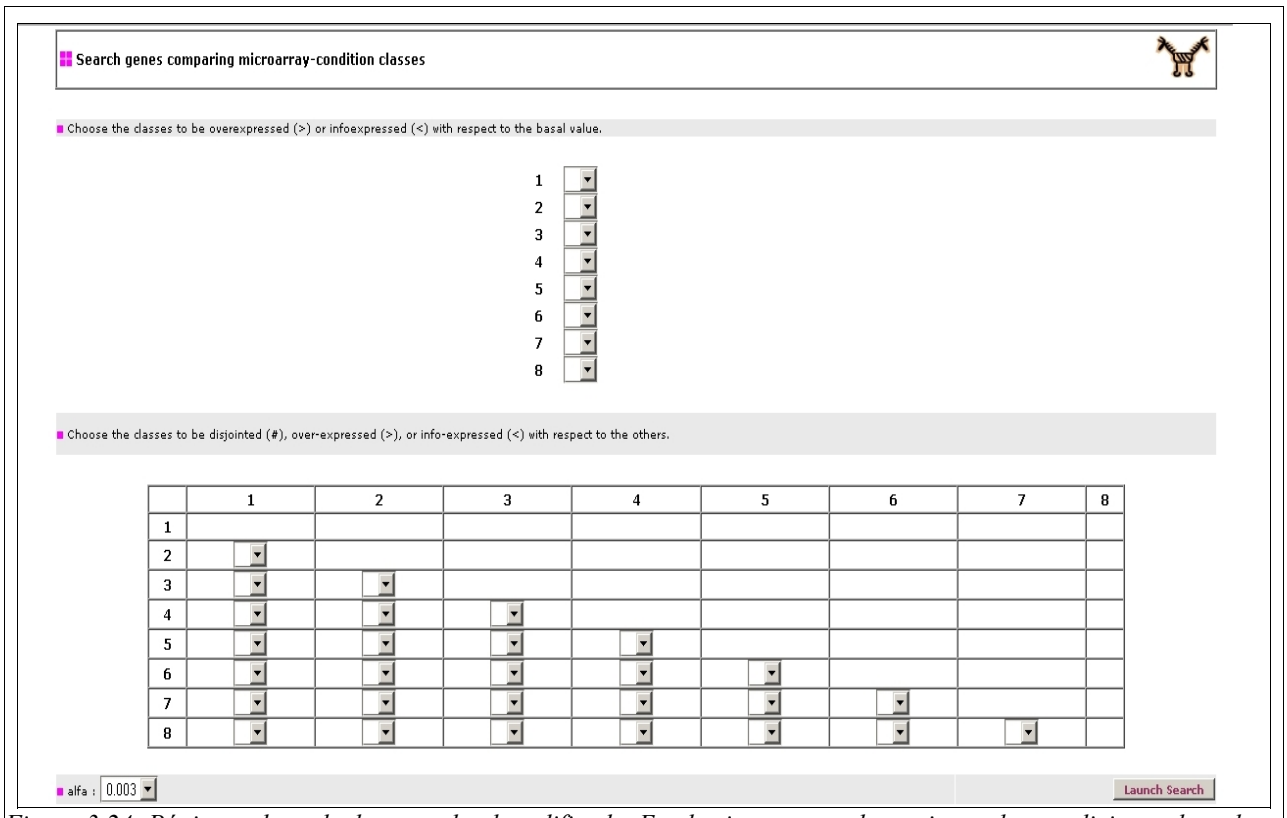
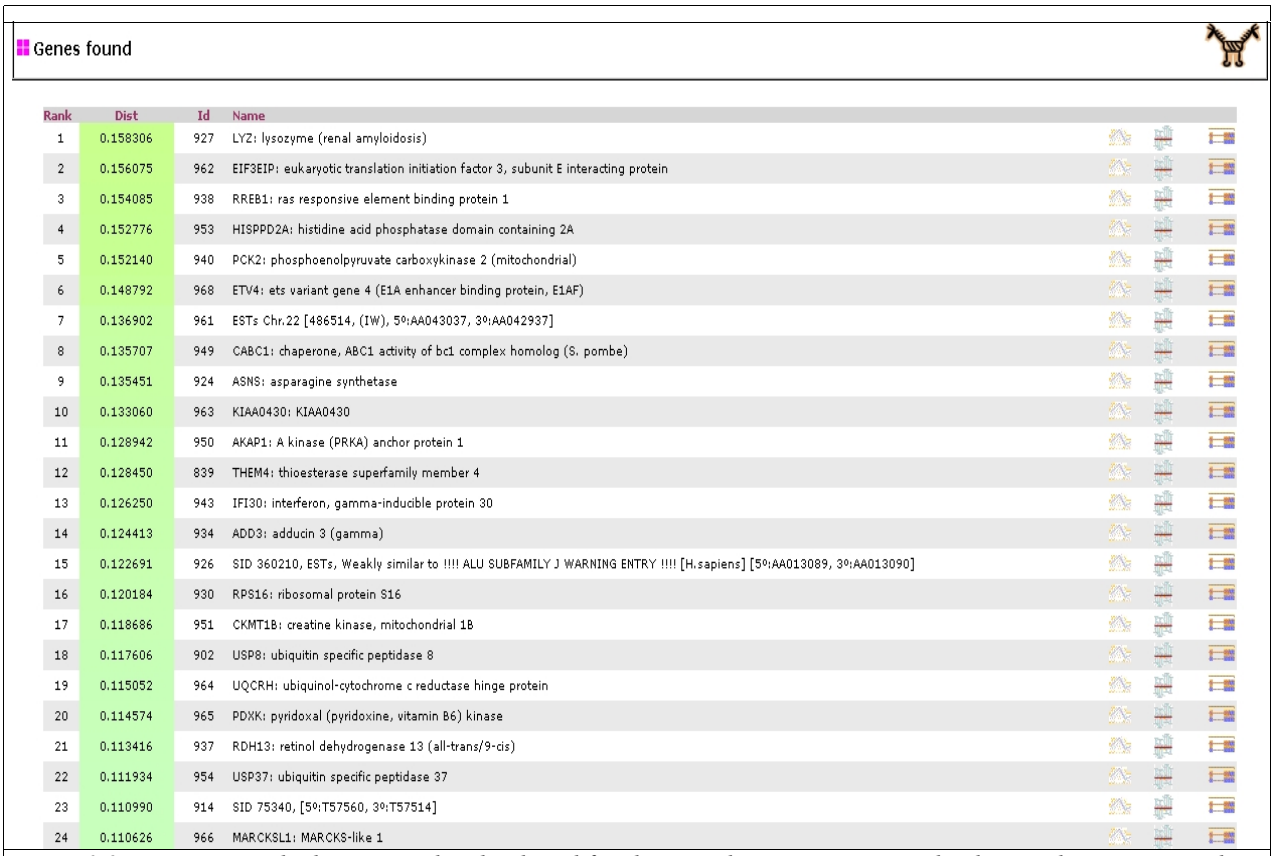


Figura 3.24: Página web appl_classegs.phtml modificada. En el primer apartado se ajustan las condiciones de cada clúster respecto el valor basal 0 , las opciones son '<' info expresado o '>' sobre expresado. En el segundo apartado se ajustan las condiciones de cada clúster respecto el resto, las opciones son: < info expresado respecto al clúster correspondiente a la columna, > sobre expresado respecto al clúster correspondiente a la columna y # sin solapamiento. Mediante el listado de la variable alfa se selecciona el nivel de confianza (0.003-99.7%, 0.009-99.1%, 0.049-95.1%). Para realizar la búsqueda de los genes se habilita el botón "Launch Search". Hay que tener en cuenta que las condiciones solo se validan para los clúster de la plantilla actual, es decir, si la plantilla actual tiene 5 clústers las condiciones para los clústers mayor de 5 no se tienen en cuenta.



Genes found

Rank	Dist	Id	Name
1	0.158306	927	LYZ: lysozyme (renal amyloidosis)
2	0.156075	962	EIF3EIP: eukaryotic translation initiation factor 3, subunit E interacting protein
3	0.154085	938	RREB1: ras responsive element binding protein 1
4	0.152776	953	HISPPD2A: histidine acid phosphatase domain containing 2A
5	0.152140	940	PCK2: phosphoenolpyruvate carboxykinase 2 (mitochondrial)
6	0.148792	968	ETV4: ets variant gene 4 (E1A enhancer binding protein, E1AF)
7	0.136902	961	ESTs Chr.22 [486514, (IW), 5º:AA043037, 3º:AA042937]
8	0.135707	949	CABC1: chaperone, ABC1 activity of bc1 complex homolog (S. pombe)
9	0.135451	924	ASNS: asparagine synthetase
10	0.133060	963	KIAA0430: KIAA0430
11	0.128942	950	AKAP1: A kinase (PRKA) anchor protein 1
12	0.128450	839	THEM4: thioesterase superfamily member 4
13	0.126250	943	IFI30: interferon, gamma-inducible protein 30
14	0.124413	934	ADD3: adducin 3 (gamma)
15	0.122691	926	SID 360210, ESTs, Weakly similar to !!!! ALU SUBFAMILY J WARNING ENTRY !!!! [H.sapiens] [5º:AA013089, 3º:AA013090]
16	0.120184	930	RPS16: ribosomal protein S16
17	0.118686	951	CKMT1B: creatine kinase, mitochondrial 1B
18	0.117606	902	USP8: ubiquitin specific peptidase 8
19	0.115052	964	UQCRH: ubiquinol-cytochrome c reductase hinge protein
20	0.114574	965	PDXK: pyridoxal (pyridoxine, vitamin B6) kinase
21	0.113416	937	RDH13: retinol dehydrogenase 13 (all-trans/9-cis)
22	0.111934	954	USP37: ubiquitin specific peptidase 37
23	0.110990	914	SID 75340, [5º:T57560, 3º:T57514]
24	0.110626	966	MARCKSL1: MARCKS-like 1

Figura 3.25: Pagina appl_classegsresults.phtml modificada. Lista los genes que cumplen las condiciones ajustadas por el usuario. Ordenados por la distancia obtenida en la búsqueda de mayor a menor, la columna Dist muestra esta distancia. La distancia mide el ranking para mejor candidato a ser un gen marcador, cuanto mayor sea mejor candidato. El degradado de color de la columna Dist indica si las distancias son muy parecidas, cuanto más degradado más variabilidad de distancias.

4.- Informe técnico

Muchos de los cálculos se realizaron usando librerías (paquetes) que no son estándar de R. Para usar estas librerías hay que instalarlas previamente en el sistema y después cargarlas una vez en el intérprete R usando el comando *library(nombreLibrería)*.

4.1.- Implementación R en Perl

Para RSPerl se ha de instalar previamente la librería en el sistema siguiendo los pasos que se indican en la página oficial de RSPerl[33]. A continuación se muestra un código sencillo de Perl donde se explica como implementar código R :

```
//importamos módulo R de la librería RSPerl para poder usar sus
//métodos,es importante recalcar que esto no inicializa el intérprete.
use R
// inicializamos el intérprete, donde &R es el objeto de interconexión
//entre Perl y R.
&R::initR("--silent");
//implementación de método sum() de R, guarda el resultado de la suma
//realizada por el intérprete en la variable de Perl $x
$x = &R::sum(1,2,3);
//muestra el resultado de la suma (6) por la salida estándar
print "Sum = $x\n";
```

Para usar la librería Statistics::R no hace falta instalar nada puesto que viene incluida en la instalación de R-Statistics (R). El código ejemplo que se muestra a continuación explica como podemos implementar código R en Perl:

```
	//importamos módulo R de la librería propia de R-Statistics con lo que
estaremos importando métodos de R
	use Statistics::R;
	//declaración del objeto en el cual guardaremos la instancia de R
	my($R);
	//creamos una istancia de R,es decir, creamos interconexión con R
	$R=Statistics::R->new($pathMicro);
	//inicializamos el intérprete R en el caso de que no exista otra
	//inicialización, si no es el caso, usa la ya existente. Si usamos el
	//método startR se creará siempre una nueva inicialización del intérprete.
	$R->start_sharedR ;
	//implementación del método sum().El método send manda un comando string
	//al intérprete R; para indicar el número de comandos que se van a mandar
	//se insertan tantas 'q' como comandos haya al inicio de la cadena
	$R->send(qq`x<-sum(1,2,3)\n print(x)`);
	//la salida estándar de R que generan los comandos ejecutados se almacena
	//en un búffer interno de R. Para obtener este búffer se usa el método
	//read. Esta lectura borra el contenido del búffer.
	my $x = $R->read ;
	//se muestra por salida estándar el resultado de la suma.
	print "Sum = $x\n";
	//se para el intérprete de R
	$R->stopR() ;
```

4.2.- Lectura de la microarray de entrada

Para identificar las diferentes microarrays que puedan haber guardadas en el sistema se les asigna un número entero (id_microarray).

Cada microarray se compone de 3 ficheros:

- id_microarray.genes: fichero de texto que contiene en cada linea el código del gen seguido de la descripción del mismo (“código”:”descripción”). Cada fila se separa usando el carácter de salto de linea.
- id_microarray.nsamples: fichero de texto que contiene los nombre de todas las condiciones muestrales que se realizaron para los genes. Cada fila se separa usando el carácter de salto de linea.
- id_microarray.samples: fichero de texto que contiene los valores de los niveles de expresión, cada linea corresponde a un gen y cada columna a una condición muestral. El número de lineas debe coincidir con el número de genes que haya en *id_microarrays.genes* y el número de columnas debe coincidir con el número de condiciones muestrales que contenga *id_microarray.nsamples* . Cada fila se separa usando el salto de linea y cada columna se separa usando el tabulador.

Se presupone que el formato de los fichero de entrada son correctos por lo tanto no se validan.

Para la lectura de los nombre de los genes y las condiciones muestrales se usa el método de R *readFile* y para la lectura de los niveles de expresión se utiliza el método *read.table*. La diferencia es que *read.table* está orientado a la lectura de ficheros donde los valores forman una matriz numérica mxn permitiendo ajustar una serie de opciones específica para matrices, en cambio *readFile* realiza una lectura normal de un fichero de texto.

Los parámetros de entrada para *readFile* son:

- *fileName*: string, corresponde al nombre del fichero de lectura. Si el fichero no está en el

directorio actual de ejecución el fileName debe incluir también el path donde se ubica el fichero.

Los parámetros de entrada para *read.table* son:

- *fileName*: string, corresponde al nombre del fichero de lectura. Si el fichero no está en el directorio actual de ejecución el fileName debe incluir también el path donde se ubica el fichero.
- *row.name:* vector de strings o entero, contiene los nombres de cada fila que se asignará a la matriz obtenida. También se puede definir el número de columna del fichero en el cual se ubican los nombres de las filas. En el caso de no asignar nombre *row.name=NULL.*
- *sep*: char, define el carácter que se usa en el fichero para separar columnas. "\t" para el tabulador.
- *fill*: boolean, en el caso de que haya filas con columnas vacías (huecos) llena esa posición con el valor NA

La implementación de la lectura de microarrays queda del siguiente modo:

```
//lectura de los nombres de los genes. La variable contendrán un array de
//m genes.
nameGenes=readFile(fileGeNames);
//lectura de los nombres de las condiciones. La variable contendrán un
//array de n condiciones. Como parámetro se le pasa el nombre del fichero
nameSamples=readFile(fileSampleNames);
//lectura de los valores de la microarray. El resultado obtenido es una
//variable del tipo Table con mXn valores correspondientes a los niveles
//de expresión de cada gen en determinadas condiciones muestrales
matrizDatos=read.table(filecondiciones
muestrales,row.names=NULL,sep="\t",fill=TRUE);
```

Se intentó asignar los nombres de las filas y de las columnas usando estos métodos pero por problemas internos de éstos no se pudo. Para solucionarlo se optó por realizar una conversión del tipo de datos *table* de la matriz resultante al tipo de datos *matrix*. Una vez realizada esta conversión se asignaron los nombre sin problema. Para ello se usan las instrucciones *rownames* y *colnames* :

```
//asignación de nombres a cada una de las filas. El parámetro de entrada
//es una variable del tipo matrix y la variable de asignación un vector de
//strings

rownames(matrizDatos)=nombresFilas

//asignación de nombres a cada una de las columnas. El parámetro de
//entrada es una variable del tipo matrix y la variable de asignación un
//vector de strings

colnames(matrizDatos)=nombresColumnas
```

Tal y como se explico anteriormente es necesario transponer la matriz para poder realizar el estudio sobre las condiciones muestrales. La transposición se realiza después de rellenar los posibles huecos (datos vacíos con valor NA) que hayan en la microarray. En la siguiente sección se explica como realzar la corrección de los huecos y como realizar la transposición de la microarray.

4.3.- Corrección de "celdas vacías" en la microarray de entrada

La librería *impute*[12] de R-Statistics incluye el método *impute.knn* que realiza un "rellenado" automático de estos valores vacíos.

Impute.knn usa los k vecinos más próximos en el espacio de los genes para el rellenado de los valores de expresión vacíos. Por este motivo se debe pasar como parámetro de entrada la matriz original, es decir, los genes como filas y las condiciones muestrales como columnas.

Para cada gen con valores vacíos, *impute.knn* encuentra los *k* vecinos más cercanos usando la

distancia euclidiana[i], escogiendo las columnas para los cuales ese gen tiene valores asignados. Alguno de los vecinos candidatos puede ser que tenga su coordenada vacía, en ese caso promedia la distancia de las coordenadas que no estén vacías. Una vez encontrados los *k* vecinos más cercanos del gen, calcula el valor de los elemento vacíos realizando el promedio de los valores (no vacíos) de sus vecinos.

Los parámetros que se usaron para implementar el relleno de huecos fueron los que usa el método por defecto. Son los siguientes:

- *data*: tipo matrix, matriz con elementos vacíos donde las filas corresponden a los genes y las columnas a las condiciones.
- *k*: int, número de vecinos próximos que se usa para calcular el valor de relleno. Por defecto se usan los 10 vecinos más próximos. No es recomendable usar un número elevado de vecinos ya que el valor obtenido puede alejarse de la aproximación real.
- *rowmax*: int, porcentaje máximo de elementos vacíos que puede haber en una fila. Por defecto es el 50%. En el caso de que se supere el máximo se usa la media global por condición.
- *colmax*: int, porcentaje máximo de elementos vacíos que puede tener una columna. Por defecto es el 80% de los valores. En el caso de superarse el máximo el proceso manda una excepción.

Esta instrucción retorna una clase con varios objetos. El que interesa es el objeto *data* que no es más que la microarray de entrada pero con los huecos rellenados.

La instrucción para implementar el rellenado de elementos vacíos en una matriz es la siguiente:

```
//rellenamos huecos de una matriz. El objeto data de la clase resultante
//del método impute.knn contiene la matriz con los huecos rellenados

matrizSinHuecos=impute.knn(matrizConHuecos).data
```

Una vez rellenados los huecos ya se puede realizar la transposición de la matriz para tener en las filas las condiciones y en las columnas los genes como se indicó previamente. La instrucción de R que realiza esta operación es *t()* y simplemente retorna la matriz transpuesta respecto la microarray de entrada.

```
//obtención de la matriz transpuesta

transMatriz=t(matriz);
```

Obtenida la matriz sin huecos y transpuesta se procede a la implementación de los distintos métodos de agrupación.

4.4.- Agrupación de condiciones muestrales: PC, MDS, SOM, SOTA, PAM, HC, K-MEANS

Para implementar el algoritmo *som* inicialmente se usa la libraría *som*[[5]] de R-STATISTICS.
El método de esta librería que realiza esta operación es *som()*.
Los parámetros del método *som* son los siguientes:

- *data*: matrix, matriz de datos donde las filas son las condiciones a agrupar y las columnas son los genes
- *xdim*: int, dimensión x para el mapa som
- *ydim*: int, dimensión y para el mapa som
- *radius*: int, radio inicial para el calculo de vecindad entre neuronas que se utiliza en el entrenamiento. Este valor debe coincidir con la dimensión del mapa bidimensional, por lo tanto es x*y. Disminuye linealmente hacia 1 durante el entrenamiento.

---

i) *Distancia euclidiana entre dos puntos* $(x_0,...x_n)$ *e* $(y_0,...y_n)$: $d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$

- *topol*: string, los posibles valores son “rect” y “hexa”. Define la topología del mapa bidimensional. Se usa el valor por defecto “rect”.
- *Neigh*: string, los posibles valores son “bubble” y “gaussian”. Define la función de vecindad que se usa en el algoritmo. Se usa el valor por defecto “gaussian”.

Un ejemplo de implementación del agrupamiento de una microarray en 8 clústers sería:

```
somObj=som(microArray,xdim=2,ydim=4,radius=8,topol="rect",neigh="gaussian");
```

En este caso se puede observar que para obtener 8 clúster podemos configurar el mapa de 4 maneras distintas. Dos serían x=1 e y=8 y a la inversa x=8 e y=1 ,y las otras dos serían x=2 e y=4 y a la inversa x=4 e y=8. Pues bien, después de realizar varios tests y analizar los resultados(ver *Anexo 1*) se pudo observar como la agrupación entre las combinaciones 1x8 y 8x1 eran iguales y lo mismo para las combinaciones 2x4 y 4x2 , ahora bien, la agrupación era distinta entre los dos pares de combinaciones.

El valor de retorno es un objeto de la clase *som*. Esta clase contiene varios atributos, los nombres de los atributos de una clase en R se pueden obtener usando el comando *names(obj)*. De todos los atributos del objeto *som* interesa el atributo *visual*.

El atributo *visual* contiene el mapa de asignaciones de cada una da las condiciones. Para acceder a este atributo basta con usar el comando *obj$nombreAtributo.* A continuación se muestra el resultado de la agrupación para x=2 e y=4 de una microarray de 118 condiciones muestrales(*objSom$visual*):

// las diferentes combinaciones de x e y corresponden a un clúster, en este caso tendremos los 8 clústers: 00, 01,02,03,10,11,12 y 13. qerror es el error cuadrático de la distancia entre el vector de observación y el vector de código que tiene asignada cada neurona.

	x	y	qerror
1	0	1	4.816948
2	0	3	4.349854
3	1	1	4.630845
4	0	0	4.537127
5	0	0	5.185083
…	…	…	...
118	?	?	?

Para acceder a cada una de las posiciones del atributo *visual* se usa el siguiente comando:

objSom$visual[idFila, (idColumna | colIni:colEnd)]

Por ejemplo si queremos obtener las coordenadas en el mapa bidimensional de la condición muestral 5 :

```
>objSom$visual[5,1:2]
      x     y
5     0     0
```

Para esta agrupación de condiciones muestrales *som* la condición 5 pertenece al clúster 0.

A continuación se describe la implementación de los algoritmos de agrupación a partir de la librería *clValid.*

clvalid puede realizar la agrupación de una microarray usando los siguientes algoritmos: kmeans, diana, fanny, som, model, sota, pam, clara, y agnes. Sobre cada uno de los resultados que se obtienen de aplicar los métodos de agrupación, *clvalid* realiza tres validaciones de integridad distintas, en la siguiente sección se detallaran cada uno .

Por lo tanto, vista la amplia versatilidad de la librería *clvalid* se optó por usarla para el cálculo de los algoritmos de agrupación: som, sota, kmeans, pam y hc.

Los parámetros de entrada son:

- *data*: matrix, microarray con las condiciones como individuos y los genes como variables
- *nClust*: int o vector, número de clústers para los cuales se calculará la agrupación. Puede ser solo un valor para k o un rango de k's . El valor mínimo es 2.
- *clMethods*: vector string, vector con todos los nombres de los métodos a usar.

- *maxitems*: int, número máximo de filas a clasificar. Este valor debe coincidir con el número de codiciones experimentales de la microarray dada.
- *validation:* string, nombre del tipo de validación a calcular para cada agrupación. Se detallará en la siguiente sección.
- *metric*: string, puede ser “euclidean”, “manhatan” o “correlation” y corresponden a la métrica usada para el cálculo de la matriz de distancias. Se escoge la métrica por defecto “euclidean” después de realizar una serie de tests descritos en la *sección 3.4.4* .

Para agrupar por ejemplo una microarray dada de 118 condiciones en 2,3,4, …, 8 clústers usando los algoritmos *som* y *hc* se implentaría con el siguiente comando:

```
clValid(data=microArray,nClust=2:8,clMethods=c('som','hierarchical'),maxitems=11
8,validation="internal");
```

El resto de parámetros se dejan los que vienen por defecto pudiendo variarlos en un futuro estudio.

Este método retorna un objeto de la clase *clvalid*. Esta clase de divide internamente en *slots*, donde cada slot representa un objeto de la clase. Para acceder a estos slots hay dos comandos:

slot(objClvalid,'nameSlot')

objClvalid@'nameSlot'

Los slots que interesan son *measures* y *clusterObjs*. El primer slot contiene todos los cálculos de integridad para cada unos del las agrupaciones y el segundo slot contiene los objetos correspondientes a cada una de las agrupaciones.

En el caso del ejemplo de antes tendríamos un objeto para *som* y otro para *hierarchical*. Accederíamos a cada uno de ellos a partir del nombre del objeto:

```
objClvalid@'clusterObjs'$som

objClvalid@'clusterObjs'$hierarchical
```

Otro comando más intuitivo para obtener una agrupación de condiciones muestrales concreta es *clusters()*. Se implementa fácilmente pasándole por parámetro la variable resultante de *clValid* y el nombre del algoritmo del cual queremos obtener sus resultados. Por ejemplo :

```
> somObjs=clusters(objClvalid,'som')
```

Todos los métodos de agrupación excepto el *hierarchical* retornan una lista donde cada posición de la lista corresponde a un objeto de la clase del método, habrá tantos objetos como clústers se hayan calculado. Estos objetos se identifican por el número de clústers que contiene la agrupación.

Para el ejemplo anterior tenemos:

```
> names(somObjs)

[1] "2" "3" "4" "5" "6" "7" "8"
```

Si se quiere acceder a la agrupación de 4 clústers por ejemplo:

```
>somObjs$'4'
som map of size 1x4 with a rectangular topology.
Training data included.
```

Se puede observar como retorna un objeto de la clase *som* que tendrá sus propios atributos. Cada método retorna sus propios objetos con sus propios atributos.

Para el obtener la agrupación de un número concreto de clústers para el método *hierarchical* se usa el comando *cutree()*. Este comando obtiene la agrupación de las condiciones cortando el árbol resultante del algoritmo en k clústers. En la *fig.4.1* tenemos un dendrograma ejemplo de una agrupación de 5 condiciones muestrales. A continuación se muestra como obtener la agrupación de las condiciones muestrales en 2 y 3 clústers a partir del dendrograma resultante de aplicar la agrupación *hc*.

```
> cutree(hcObj,2)
```

```
[1] 1  2  1  1  2  1
> cutree(hcObj,3)
[1] 1  2  1  1  2  3  //cada posición del array corresponde a la condición muestral de la microarray y el su valor es el identificador del clúster
```

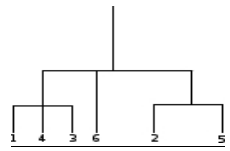


Figura 4.1: Dendrograma resultante del algoritmo hierarchical para 6 condiciones muestrales.

Pasaremos a describir para el resto de métodos como obtener las asignaciones de cada una de las condiciones a un clúster determinado una vez escogido el número de clústers. Estos son los nombres de los atributos para las distintas clases de objetos que contienen esta información:

· som: *unit.classif*

· sota: *clust*

· kmeans: *cluster*

· pam: *clustering*

Para acceder a estos atributos:

objagrupación$nameAtributo.

Por ejemplo, para obtener la agrupación de 8 clústers :

```
>somomObj$'8'$unit.classif
[1]1 2 1 1 2 3 4 5 5 6 6 7 7 8 8 ...
kmeansObj$'8'$cluster
[1]1 1 5 1 2 3 4 3 2 1 6 2 7 8 1 ...
sotaObj$'8'$clust
[1]1 2 1 1 2 3 4 5 3 1 6 1 7 8 3 ...
pamObj$'8'$agrupación
[1]1 2 8 8 2 3 4 5 5 6 6 7 7 5 5 ...
```

El formato de los datos de retorno es el mismo que para el método *cutree(), r*etornan un array donde cada posición corresponde a una condición y su valor el identificador de la clase a la cual pertenece.

·Implementación Multidimensional Scaling (MDS)

Para el escalado MDS se usó el comando *cmdscale*. Este método reduce la dimensión de las variables de la microarray.

El comando *cmdscale* precisa como microarray de entrada la matriz distancia de la microarray. Esta matriz nueva se calcula usando el comando *dist*(). Calcula la distancia euclidiana entre cada par de condiciones muestrales de la microarray retornando una matriz donde cada posición es la relación entre cada par de condiciones y el valor la distancia euclidiana. Su implementación es la siguiente:

```
>dist(microarray)
        1       2       3       4       5       6       7 ...
2  10.049876   //distancia euclidiana entre la condición muestral 2 y la condición  1
3   2.000000 10.049876
4  20.223748 10.198039 20.024984
5   4.000000 10.440307  2.000000 20.024984
. . .
```

Finalmente la implementación del escalado MDS queda del siguiente modo:

```
>cmdscale(disMatrix,k=6,eig=TRUE)
```

El parámetro *k* es el número máximo de dimensiones que se calcularán para la matriz. Se escogió 6 después de consultar a Juan Antonio Cedano que ya había realizado varios tests relacionados con el MDS.

El parámetro *eig* es importante activarlo ya que indica si se calculan o no los valores propios o eigenvalues[i] para cada dimensión *k*. Estos eigenvalues son los que se usan para encontrar la dimensión *k* óptima.

Para encontrar esa dimensión óptima que reduzca al máximo la información pero que a su vez se pierda la menos cantidad posible se implementa con la fórmula siguiente:

```
k=min( sum(mdsMicro$eig>0),sum( cumsum(abs(mdsMicro$eig))/sum(abs(mdsMicro$eig))
<=0.95));
```

Se escoge la *k* mínima de entre el número total de eigenvalues que son mayores a 0 y del número total de promedios de los eigenvalues que superen un porcentaje dado. En este caso se escogió el 95% que también es un valor aconsejado por Juan Antonio Cedano después de realizar varios estudios.

En el *Anexo 2* se muestra un ejemplo de la implementación.

·Implementación Principal Components (PC)

Para usar el método de escalado PC se usó la librería *pcaPP*. Esta librería busca las 2 componentes principales de un conjunto de variables dado.

El método que se usó de esta librería fue PCAg*rid*. Se dejaron todos los parámetros por defecto.

En el *Anexo 2* se muestra un ejemplo de esta implementación.

El atributo *scores* del objeto de la clase pcaPP contiene los valores de las componentes principales para cada condición muestral. Estos valores son combinaciones lineales del conjunto de variables de la microarray y son las componentes que mejor resumen esta información.

4.5.- Cálculo de la integridad de las distribuciones de clústers

·Calinski-Harabasz y Hartigan

La librería *clusterSim* incluye el cálculo de estos índices a parte de otros.

El índice *Calinski-Harabasz* se calculó usando el método *index.G1().* Los parámetros de entrada son:

- *data*: matrix, matriz de datos de la microarray que se usó para la agrupación
- *cl*: vector enteros, vector que identifica a que clúster pertenece cada condición muestral de la microarray.

El valor de retorno es un número decimal. La mejor agrupación (número de clústers óptimos) será aquella que maximice este valor.

A continuación se muestra un ejemplo de la implementación:

```
> cl8    //agrupación en 8 clusters de una microarray
  [1] 4 3 4 4 4 4 2 2 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 5 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 8 5 4 4 5 5 5 7 7 7 7 7 5 5 7 5 5 7 7 7 7 7 6 6 6 6 6 6 6 6 7 7 6 5 6 5
 [75] 6 6 6 6 6 4 4 3 3 3 3 3 3 3 4 3 3 8 7 8 7 7 7 3 7 8 8 8 8 8 8 8 8 4 3 5
[112] 1 4 4 4 4 2 4
```

i) *En álgebra lineal, los vectores propios, autovectores o eigenvectores de un operador lineal son los vectores no nulos que, cuando son transformados por el operador, dan lugar a un múltiplo escalar de sí mismos, con lo que no cambian su dirección. Este escalar λ recibe el nombre valor propio, autovalor, valor característico o eigenvalue*

```
> index.G1(microarray,cl8)
[1] 19.03856
```

Para la implementación del cálculo índice *Hartigan* de una agrupación de *k* clústers se usó el método *index.H*. El indice *Hartigan* de una determinada agrupación de *k* clústers se obtiene a partir del vector de asignaciones de esta agrupación y del vector de agrupaciones de *k+1* clústers.

Se usaron los siguientes parámetros:

- *data*: matrix, matrixz de datos de la microarray que se usó para la agrupación
- *clAll*: dos vectores de enteros que indican el clúster al cual pertenece cada condición en la agrupación de todas las condiciones en *k* y *k+1* clústers.

El valor de retorno es un número decimal. El número de *k* óptimo es la *k* más pequeña (>1) para la cual el valor del índice sea ≤10.

Implementación ejemplo:

```
> cl24 //agrupación para k=8
 [1] 3 7 4 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 2 2 2 2 2 2 2
[38] 2 8 4 6 8 6 2 4 8 8 7 8 8 6 4 8 8 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 6
[75] 8 8 8 8 8 7 3 7 7 7 7 7 7 7 7 7 7 7 7 8 7 7 7 7 7 7 7 7 7 5 3 6
[112] 1 5 3 5 5 1 3
> cl19 //agrupación para k=9
 [1] 5 7 5 2 2 2 1 1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1
[38] 1 9 4 6 7 5 3 5 9 8 9 9 9 6 5 9 8 7 9 9 9 9 9 8 8 9 9 9 9 9 8 9 9 9 7 7 6
[75] 8 9 9 9 7 9 7 4 9 8 9 9 9 9 9 7 7 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 6 5 5
[112] 1 6 4 6 5 1 4
> call=cbind(cl24,cl19) //unión de las dos clasificaciones, k y k+1
   [,1] [,2]
1     3    5
2     7    7
3     4    5
4     1    2
5     1    2
…    …    …
118   3    4
> index.H(micro$trans,call) //índice Hartigan para k=8
[1] -22.18397
```

·Connectivity, Silhouette y Dunn

Para que el método *clValid()* calcule estos índices de integridad basta con asignar al parámetro de entrada *validation* el string “internal”.

El objeto resultante de *clValid* contiene un atributo que se llama *measures*, este atributo es un array de 3 dimensiones donde la primera dimensión corresponde al índice de integridad, la segunda al número de clústers y la tercera al algoritmo de agrupación.

A continuación se muestra un ejemplo de implementación:

```
//creamos agrupación som y hierarchical de 4 y 5 clústers
>res=clValid(pcMicro,4:5,clMethods=c("som","hierarchical"),
      validation="internal",118,metric="euclidean")
> integridad==measures(res)
, , som

                4        5
Connectivity 26.17698413 29.8575397
Dunn          0.06848316  0.1010114
```

```
Silhouette    0.39975522  0.4398218

, , hierarchical

                4         5
Connectivity 18.73095238 24.30515873
Dunn          0.09312712  0.09312712
Silhouette    0.38967368  0.38120495
```

```
>integridad["Dunn","5","som"]
[1] 0.1010114
```

El parámetro *metric* del método *clValid()* define el algoritmo que se usa para calcular la matriz distancia que *clvalid* usa tanto para la agrupación como para el cálculo de los índices de integridad.

Estas métricas son: *euclidean, manhattan* y *correlation*.

Se escoge la euclidean por los test explicados en las sección 3.4.4.

Ejemplo de implementación con *clValid* :

```
<
res1=clValid(pcMicro,2:15,clMethods=c("kmeans","sota","som","pam","hierarchical"
),validation="internal",118,metric="euclidean")
```

4.6.- Biclustering

Se implementaron los algoritmos ISA (Iterative Signature Algorithm), CC (basado en el trabajo de Cheng and Church) y Bimax.

Todos estos algoritmos están basados en el *biclustering*. Este algoritmo clasifica simultáneamente las filas y las columnas de la microarray de entrada.

Para implementar ISA se usó la librería *isa2*[10]. A continuación se muestra un ejemplo:

```
//creamos biclustering usando el algoritmo isa2
< isaObj=isa(microArray)
//conversión objeto ISA a objeto estándar bicluster
< bcIsa=isa.biclust(isaObj)
```

Para la implementación de CC y Bimax se usó la librería estándar *biclust*[8]. Mediante un parámetro del método *biclust()* se puede seleccionar un método u otro.

Para implementar el CC se usaron los siguientes parámetros del método *biclust*:

- *method*: BCCC(), método que calcula el *biclustering*
- *delta:* decimal, puntuación máxima de aceptación, 0.15 en este caso
- *alpha:* decimal, factor de escala, 1 en este caso
- *number*: int., número de biclusters. En este caso solo interesa 16 como máximo

La implementación queda del siguiente modo:

```
//creación objeto estándar bicluster a partir de algoritmo BCCC()
< bcCc=biclust(microArray,method=BCCC(),delta=0.15,alpha=1,number=16);
```

Para implementar el *Bimax* se usaron los siguiente parámetros del método *biclust:*

- *data:* matriz lógica que representa a la microarray.
- *method*: BCBimax(), método que calcula el *biclustering*

- *minr:* int, número mínimo de filas del bicluster resultante. Se fija 4 por recomendación de la propia librería
- minc: int, número mínimo de filas del bicluster resultante. Se fija 4 por recomendación de la propia librería
- *number*: int., número de biclusters. En este caso solo interesa 16 como máximo

Para obtener la matriz lógica de la microarray se usa el comando *binarize(microarray).*

Implementación:

```
//calculo matriz binaria(lógica) de la microarray
< binMicro=binarize(microArray)
//creación objeto estándar bicluster a partir de algoritmo BCBimax()
< bcBimax=biclust(binMicro,method=BCBimax(),minr=4,minc=4,16)
```

4.7.- Tratamiento de las condiciones muestrales outlayers

Se consideran condiciones muestrales outlayers a las condiciones muestrales que no tienen un clúster asignado o aquellas que pertenecen a un grupo que no contiene más de un número determinado de condiciones muestrales.

Se determinó que cada clúster como mínimo debía contener el 5% de las condiciones muestrales a agrupar. Por ejemplo, si se quieren agrupar un total de 118 condiciones muestrales los clústers como mínimo deberán contener 6 condiciones muestrales (5%), todas las condiciones muestrales que pertenezcan a un clúster que no cumpla esta condición se las denomina outlayer.

Supongamos que para una agrupación de *k* clústers tenemos un grupo con condiciones muestrales outlayers. El algoritmo queda del siguiente modo:

- Buscar a que clústers pertenecían estas condiciones muestrales para la agrupación *k-1.*
- Para cada clúster seleccionar todas las condiciones muestrales pertenecientes a este clúster para agrupación *k-1*.
- Buscar los clústers a los que pertenecen las condiciones muestrales seleccionadas para agrupación *k.*
- Escoger el clúster que mayor número de condiciones muestrales contenga para agrupación *k.*
- Reasignar las condiciones muestrales outlayers a este clúster.

4.8.- Normalización identificadores clústers

Este procedimiento se lleva a cavo siempre que se haya realizado una fusión de condiciones muestrales y antes de guardar los ficheros correspondientes a las plantillas de colores.

Al reubicar condiciones muestrales outlayers en otros clústers (fusión de clústers) estamos eliminando grupos por lo que los ids de los grupos contendrán saltos. Supongamos que tenemos una agrupación de k=5 con ids 1,2,3,4,5 y tenemos que las condiciones muestrales del clúster 2 son outlayers y que por el proceso de fusión reubicamos las condiciones muestrales del clúster 2 al clúster 3. Ahora tendremos los ids 1,3,4,5 para una agrupación de k=5 fusionada. Como por especificación las plantillas no pueden contener saltos se reasignan los ids del siguiente modo:

- El id 1 sigue siendo el mismo
- El id 3 pasa a llamarse 2
- El id 4 pasa a llamarse 3
- Y el id 5 pasa a llamarse 4

Por lo tanto, para este agrupación de k=5 fusionado tenemos que su *k* real es 4 y los ids son

1,2,3,4.

Lo mismo pasa con el filtrado de outlayers. Este caso sucede cuando no se han podido reubicar los outlayers a otros clústers. El filtrado elimina estas condiciones muestrales de la plantilla y aparecen huecos entre los ids de los clústers de manera que hay que normalizar como se explicó antes.

4.9.- Integración de la agrupación en el preproceso

El preproceso se lanza a través de un programa ejecutable programado en lenguaje C. Este programa se ubica en el directorio */var/www/cgi-bin/pcop/fullcorrelations/* del servidor y se llama *lanzadora.cc .*

Como que el tiempo de ejecución de este preproceso es muy elevado se decide realizar un simulador de preproceso en el cual solo el subproceso correspondiente a la agrupación de condiciones muestrales fuera el que estuviera activo. Se le llamó *lanzadoraPruebaHtmlR.cc.*

En el método *main()* de este fichero se incluye la instrucción *goto* que realiza un salto a la parte del código que le indiquemos, en este caso se hace un salto a la parte del código que realiza todos los cálculos de agrupación.

En esta parte del código se realizan dos acciones o procesos. Uno consiste en crear el directorio "*Colors_Historic*" *(sección 4.10) y* el otro en realizar la llamada a un script de Perl *(clusteringSamplesR.pl)* que es el que se usa para conectar con el intérprete de R. El intérprete R es quien realiza todos los cálculos de agrupación a partir de los algoritmos de agrupación implementados en lenguaje R a parte de guardar todos los resultados en ficheros.

Todos los scripts relacionados con el desarrollo de la aplicación se ubican en el directorio */var/www/cgi-bin/pcop/fullcorrelations/* tanto los de Perl como los de R.

El script de Perl *clusteringSamplesR.pl* realiza principalmente dos tareas:

- Cargar en el intérprete de R el código del programa principal implementado en R *(clsPreprocess.r)* y ejecutarlo.
- Normalizar todos los fichero resultantes obtenidos tanto del directorio base *RClustering_Samples* como del directorio *Best (sección 4.10)*

El script *clsPreprocess.r* es el programa principal implementado en R que realiza todos los cálculos de agrupación y registro de resultados.

Antes de emular el preproceso se tuvo que instalar todas las librerías de R necesarias. En este proceso de instalación se encontró un error de incompatibilidad. No se pudo instalar la librería *clusterSim* que era la encargada de calcular el índice de integridad *Hartigan.* Esta librería era incompatible con la versión de Linux del servidor. Se prescindió de ella, por lo tanto no se calcula el índice *Hartigan* (solo se calculan el índice *Dunn y Silhouette* de la librería *clValid*) .

Para emular todo el preproceso simplemente era necesario ejecutar el script *lanzadoraPruebaHtmlR* del sigiuiente modo:

```
// lanzar el preproceso para la microarray con identificador 17
[user@server]$ ./lanzadorPruebaHtmlR 17
```

Para emular este script vía web se implementó un programa de pruebas desarrollado en PHP (*http://revolutionresearch.uab.es/applic/gexp/microarray/clusterSamples.php*). Este script permitía lanzar el preproceso para una microarray solicitada por un usuario a través de un formulario web.

4.10.- Gestión de resultados de la agrupación

Todas las microarrays cargadas en el sistema se ubican en el directorio */var/www/cgi-bin/pcop/microarray/*.

En este directorio, cada microarray, tiene su propia carpeta. Estas carpetas se nombran a partir del identificador propio de cada microarray, "*mId" (m17,m15,etc …).* En cada una de estas carpetas se guardan

todos los directorios y ficheros propios de cada microarray.

Una vez se ejecuta el subproceso de agrupación sobre una microarray determinada se generan una serie de ficheros y directorios que contienen todos los resultados obtenidos.

Los ficheros más relevantes de todos son los que tienen la extensión *.colors* y se les conoce por plantilla de colores .

4.10.1.- Ficheros clústers

Estos ficheros que tienen un formato específico son los que el usuario final podrá usar en la aplicación web *(PCOPGene*[14]*)* para sus propias investigaciones.

Se les conoce por plantillas de colores ya que se usan para poder identificar en una representación gráfica cada uno de los clústers de condiciones muestrales pintando cada uno con un color distinto.

Los ficheros *.colors* contienen una agrupación determinada de todas las condiciones muestrales de una microarray determinada. Cada fila del fichero corresponde a una relación IdCluster – IdMuestra.

Por limitaciones técnicas el límite de clústers permitido en un fichero de clústers es de 8 aunque la aplicación está parametrizada para poder generar más clústers ya que en futuros trabajos esta limitación se corregirá. Se pueden guardar plantillas de hasta 9 clústers pero no se tendrá en cuenta el clúster 9 de estas plantillas.

Otra especificación de estas plantillas es que no pueden haber *outlayers.* Este tipo de condiciones muestrales se eliminan del fichero. Por lo tanto, si una determinada agrupación contiene *outlayers* en el fichero solo estarán las condiciones muestrales que no sean *outlayers.*

Finalmente otra de las particularidades de las plantillas de colores está relacionada con el identificador del clúster. Las condiciones muestrales se ordenarán de modo ascendente a partir de sus identificadores del clúster. Los identificadores de clúster deben ser correlativos y sin huecos, es decir, si por ejemplo tenemos 4 clústers los identificadores serían 1,2,3,4 y no 1,3,4,5.

A continuación se describen todos los directorios y ficheros generados .

4.10.2.- Directorios y ficheros

En el directorio raíz de la microarray *(/var/www/cgi-bin/pcop/microarray/mId/)* clasificada se crean los siguientes directorios y ficheros:

- *mId.samples_full :* fichero.
- *Colors_Historics :* directorio
- *RClustering_Samples* : directorio.

mId/mId.samples_full

En el caso de que existan huecos en la microarray se genera este archivo donde los huecos se habrán rellenado. Se guarda para futuras investigaciones.

Por ejemplo, si existen huecos en la microarray 17 se crea el fichero 17.samples_full donde esos huecos se han rellenado.

mId/Colors_Historic/

Este directorio inicialmente no contendrá ningún archivo. Está destinado a guardar todas las plantillas de colores de cada uno de los usuarios web que hayan decidido guardar a través de la aplicación web PCOPGene.

Los nombres de todos los ficheros ubicados en este directorio deben contener tanto el identificador de la microarray como el identificador de usuario ya que estos ficheros se usan en la aplicación web PCOPGene donde se a identificado previamente el usuario.

El nombre de cada una de las plantillas guardadas tiene el siguiente formato:

IdMicro_IdUser_nameFile.colors

Por ejemplo, si el usuario con identificador 202 guarda un fichero de clústers con el nombre "defaultColors" correspondiente a la microarray con identificador 17 el nombre del fichero guardado quedaría del siguiente modo:

17_202_defaultColors.colors

En este directorio también existen una serie de ficheros resultantes de un proceso de normalización.

mId/RClustering_Samples/

Es en este directorio donde se guardan todos los resultados obtenidos al aplicar el subproceso de agrupación sobre una microarray determinada.

Los ficheros generados directamente en este directorio son: ficheros de estadística, ficheros de los dos métodos de escalado (mds y pc) y todas las plantillas de colores de las distintas implementaciones de los algoritmos de agrupación.

Los directorios creados directamente en este directorio son : Best y Datos_Originales.

Ficheros estadística

Son ficheros de texto que contienen información estadística de los resultados obtenidos.

Existe un ficheros de estadística para cada uno de los algoritmos de agrupación implementados.

El nombre de estos ficheros sigue el siguiente formato:

mId_estadistica_pc|mds|ø_clusteringName_descarte|ø.txt

Tal y como se indicó anteriormente los algoritmos de agrupaciones implementados para la microarray escalada usando las componentes principales y el escalado multidimensional son : hierarchical, kmeans, som, sota y pam.

Para la microarray sin escalar se implementan los algoritmos de agrupación som, sota, pam y hierarchical.

Finalmente se distingue entre dos tipos de agrupaciones. Uno al cual se le aplica un proceso de fusionado de clústers con tal de eliminar la posible aparición de *outlayers* y otro al cual no se le aplica ningún proceso posterior al agrupación. Al segundo se le distingue con el nombre *descarte* ya que todas las condiciones muestrales outlayers se descartan o eliminan del fichero de clústers final.

Ejemplos:

17_estadisitca_pc_hierarchical.txt : estadística *hierarchical clustering* con fusión de la microarray 17 escalada usando las componentes principales.

17_estadisitca_hierarchical.txt : estadística *hierarchical clustering* con fusión de la microarray 17.

17_estadisitca_mds_hierarchical.txt : estadística clustering *hierarchical* con fusión de la microarray 17 escalada usando e escalado multidimensional.

17_estadisitca_pc_som_descarte.txt : estadística clustering *som* sin fusión de la microarray 17 escalada usando las componentes principales.

La información que contiene el fichero de estadística es la siguiente:

- Tabla integridades (tantas filas como K Reales se hayan calculado, de 4 a 16 en nuestro caso):
  - K Real: es el número de clústers en que se agrupan las condiciones muestrales
  - K Nueva: es el número de clústers final después de aplicar tanto el proceso de fusionado como el proceso de eliminado de *outlayers.*
  - Fusiones: número de clústers que se han fusionado (reubicación condiciones muestrales)

- Outlayers: número de clústers outlayers que se han eliminado (clústers que no tienen un mínimo de condiciones muestrales determinadas)
- Calinsky: valor del índice de integridad Calinsky, no se calcula (siempre valor 0)
- Hartigan: valor del índice de integridad Hartigan, no se calcula (siempre valor 0)
- Dunn: valor del índice de integridad Dunn
- Silhouette: valor del índice de integridad Silhouette

- Sección BEST:
  - Esta sección del fichero se usa específicamente para mostrar la información de las mejores K's según los índices de integridad Dunn y Silhouette a través de la web.
  - En el caso de que coincidan las K óptimas para los dos indices de integridad solo se seleccionará una de ellas, Dunn en este caso.
  - Se guardan los valores de las K's óptimas que se hayan encontrado:
    - K: número de clústers óptimo en que se agrupan las condiciones muestrales
    - numClusters: número de clústers final después de aplicar tanto el proceso de fusionado como el proceso de eliminado de *outlayers*. Este sera realmente el número de clústers óptimo.
    - val: número decimal que corresponde al valor del índice de integridad para el cual K es el número óptimo de clústers.
    - val.dunn/silhouette: número decimal que corresponde al valor del otro índice de integridad para esta K (para este índice no es la mejor K).
  - Sección Outlayers:
    - Esta sección es meramente informativa y solo está presente si hay alguna K con grupos de condiciones muestrales del tipo *outlayer.*
    - Se muestra la siguiente información:
      - Se muestra el número mínimo de condiciones muestrales que deben contener todos los clústers. Es necesario que haya como mínimo un 5% de condiciones muestrales en cada clúster.
      - Para cada K se guardan los clústers que no cumplan esta condición. Para ellos se guardan los identificadores de las condiciones muestrales de estos clústers por separado.

En el *Anexo 4* se pueden ver algunos ejemplos de la información que se guarda en este tipo de fichero.

Ficheros escalado

Son dos ficheros de textos que corresponden cada uno de ellos a los dos métodos de escalado que se realizan sobre la microarray.

El formato del nombre es el siguiente: mId.pc y mId.mds

Por ejemplo para la microarray con identificador 17 sería : 17.pc y 17.mds

Estos ficheros se guardan a nivel informativo para futuros trabajos que se quieran realizar.

El fichero correspondiente al PC guarda en formato texto una matriz de valores que corresponden a las dos componentes principales de la microarray donde tiene tantas filas como condiciones muestrales tenga la microarray.

El fichero correspondiente al MDS guarda en formato texto una matriz de valores que corresponden a cada una de las dimensiones óptimas encontradas por este algoritmo sobre la microarray donde tendrá

tantas filas como condiciones muestrales tenga la microarray.

Ficheros agrupación

Para cada algoritmo de agrupación se guarda el fichero de clústers correspondiente a cada una de las K's (número de clústers usados para agrupar las condiciones muestrales) calculadas .

En nuestro caso se usan las K's: 4,5,6,7,8,9,10,11,12,13,14,15,16

Hay que tener en cuenta que tenemos tres tipos de implementaciones de los diferentes algoritmos de agrupación. Los algoritmos de agrupaciones aplicados directamente a la microarray, los algoritmos aplicados a la microarray escalada usando PC y finalmente los algoritmos aplicados a la microarray escalada usando MDS.

También se distinguirá entre las agrupaciones que a posteriori se les aplique el proceso de fusión para eliminar posibles outlayers. A las agrupaciones que no se les aplique este proceso se les diferencia llamándolos *descarte.* Estas agrupaciones con descarte son susceptibles de contener outlayers.

En las plantillas de colores no se guardan las condiciones muestrales outlayers.

El formato queda del siguiente modo:

mId_pc|mds|ø_clusteringName_descarte|ø_K.colors

Por ejemplo:

17_hierarchical_descarte_4.colors : clustering hierarchical sin fusión de 4 clústers sobre la microarray 17 con eliminación de condiciones muestrales outlayers en el caso de que haya.

17_mds_kmeans_12.colors : agrupación kmeans con fusión de 12 clústers sobre la microarray 17 escalada usando MDS con eliminación de condiciones muestrales outlayers en el caso de que haya (posiblemente se hayan fusionado a otros clústers).

Hay que recalcar que la K que se muestra en el nombre de la plantilla no es número final de clústers en que se agrupan las condiciones muestrales ya que o por el fusionado o por la eliminación de outlayers el número de clústers K a podido disminuir. Esta información se guarda en el fichero de estadísticas.

mId/RClustering_Samples/Best

En este directorio se guardan solo las plantillas de colores que correspondan a las K's óptimas según los dos índices de integridad y para cada una de las implementaciones de los algoritmos de clustering. Son las mejores agrupaciones.

El formato es el siguiente:

mId_pc|mds|ø_clusteringName_descarte|ø_dunn|silhouette_K.colors

Por ejemplo:

17_pc_pam_dunn_7.colors : agrupación pam con fusión de 7 clústers sobre la microarray 17 escalada usando pc con eliminación de los posibles outlayers. Esta plantilla corresponde a la mejor agrupación para el índice de integridad *Dunn.*

Como se explicó antes la K que se muestra en el nombre de la plantilla puede no corresponder al número real de clústers puesto que se han podido fusionar o eliminar outlayers.

Las plantillas de colores que estén ubicadas en este directorio serán las que se muestren a través de la aplicación web PCOPGene.

mId/RClustering_Samples/Datos_Originales

Aquí se guardan una serie de ficheros que corresponden a objetos de R. Estos objetos se pueden cargar a posteriori en un intérprete de R. Para ello basta con usar el comando de R *load(objeto).*

Se guardan estos datos para futuros trabajos o análisis.

Los distintos objetos que se guardan son los siguientes:

- normal_clustering : objeto de la clase *clValid* resultante de aplicar los distintos algoritmos de agrupación directamente sobre la microarray.
- mds_clustering : objeto de la clase *clValid* resultante de aplicar los distintos algoritmos de agrupación sobre la microarray escalada usando MDS.
- pc_clustering : objeto de la clase *clValid* resultante de aplicar los distintos algoritmos de agrupación sobre la microarray escalada usando PC.
- Biclustering : objeto que contiene los resultados de aplicar los tres tipos de biclustering sobre la microarray (ISA, BiMax, BCCC).
- Objetos de clustering : para cada algoritmo de agrupación implementado diferenciando entre las agrupaciones aplicadas a la microarray directamente o a la microarray escalada y diferenciando las agrupaciones a los que se les aplica el proceso de fusión o los que no se guarda un objeto que contiene todos los resultados de dichas agrupaciones.

  El nombre del fichero es el siguiente:

  normal|mds|pc_clusteringName_descarte|fusion

  Estos objetos contienen varios campos de información:

  - K: vector que contiene todos los número de clústers que se usaron para la agrupación (de 4 a 16 en nuestro caso)
  - Para cada K se guarda :
    - 'k': vector de agrupación de las condiciones muestrales, a cada posición se le asigna un identificador de clúster.
    - 'k_Objs': matriz de k filas donde se guarda en número de condiciones muestrales de cada uno de los clústers

4.11.- Interfaz web: integrar los resultados de la agrupación en el aplicativo web

Tal y como se detalló anteriormente los directorios del servidor donde se ubican los ficheros que se utilizan en esta aplicación son el *Best* y *Colors_Historic.*

Esta plantilla se guarda en forma de fichero y se ubica en el directorio de cada microarray del servidor del applet. Se nombra del siguiente modo: *mId_userId.colors*

Por ejemplo, el fichero de clústers por defecto para la microarray 17 del usuario 300 se ubicaría en el directorio m17 del directorio de microarrays con el nombre 17_300.colors.

Las modificaciones se realizaron sobre el fichero appl_palette.php.

Para implementar la primera funcionalidad que se describe en la *sección 3.9* lo que se hizo es listar todos los ficheros que se ubican en el directorio */var/www/cgi-bin/pcop/microarray/mId/Best* . Para cada fichero se muestran los siguientes campos de información:

- Nombre: nombre del algoritmos de agrupación utilizado
- K: número inicial de clústers en que se agruparon las condiciones muestrales
- Clusters: número final de clústers en que se agruparon las condiciones muestrales después de posibles procesos de fusión y eliminación de *outlayers*.
- Silhouette: valor índice integridad Silhouette
- Dunn: valor índice integridad Dunn

Todos estos campos de información se extraen de los ficheros de estadística correspondientes en la sección *BEST* de estos ficheros.

Una vez se mostró este listado de ficheros se añadieron dos botones al lado de cada fichero de

clústers encontrado. Uno de ellos con la funcionalidad de actualizar el fichero de clústers actual con la plantilla seleccionada y el otro con la funcionalidad de descarga, es decir, permite al usuario descargar el fichero de clústers seleccionado en un fichero.

Para reducir la lista de plantillas se hace un filtro para eliminar agrupaciones similares.

La mayoría de casos eran las plantillas obtenidas con un mismo algoritmo de agrupación pero que se distinguían porque a unas se le aplicó el proceso de fusión y a las otras no (eliminación de *outlayers*). En el caso de que no se eliminaran outlayers ni se fusionaran clústers se obtenían dos plantillas totalmente idénticas.

Si se encuentran este tipo de plantillas solo se guarda la correspondiente a la de fusión y se descarta la otra.

Por ejemplo, si tenemos que la agrupación óptima de 17_mds_kmeans_8_silhouette.colors y el de 17_mds_kmeans_8_silhouette_descarte.colors contienen una agrupación idéntica se prescinde de la agrupación sin fusión (la de *descarte*).

4.12.- Ordenación,agrupación y normalización de los ficheros de clústers

El script donde se implementan estos procesos de mejora se llama *clsHcFiles.r.*

Para agrupar los ficheros de plantillas se usa el algoritmo Hierarchical , la librería estándar de R contiene un método que se encarga de realizar esta agrupación. El método es el *hclust()* y se le pasan los siguientes parámetros:

- Matriz de disimilitud creada a partir de una matriz de distancias.
- Method: método aglomerativo, se usa el método *complete linkage*[i] que es el que viene por defecto.

Para crear la matriz de distancia se implementa un algoritmo que mide la distancia entre cada par de plantillas de un directorio.

Cálculo de la distancia entre dos plantillas de colores :

$min(\sum_{i=0}^{K_a} min(dist(Ca_i, Cb)))$ dónde #Ca < # Cb

Ca_i- Clúster i-ésimo de la plantilla de colores A

Ca- Clústers de la plantilla de colores A

Cb- Clústers de la plantilla de colores B

Ka – número de clústers

$dist(Ca_i,Cb_j)$ - $(m \in Ca_i \wedge m \notin Cb_j) + (m \notin Ca_i \wedge m \in Cb_j)$, m número condiciones muestrales

La suma de distancias entre cada clúster de las dos plantillas debe ser la mínima y los candidatos de la plantilla B (más cercanos a un clúster de A) son únicos.

Las parejas con distancias mínimas son las candidatas para futuras asignaciones.

La distancia se calcula siempre a partir de la plantilla que contenga menos clústers.

La distancia total entre las dos plantillas es la suma de estas distancias mínimas.

Para la repetición de asignaciones candidatas se calculan de forma exhaustiva las distancias mínimas para encontrar otra combinación de asignaciones que no tengan clústers repetidos.

El método implementado que se encarga de calcular la distancia entre dos plantilla se llama

i) ***El complet linkage*** *o* ***vecino más lejano*** *es un método de cálculo de las distancias entre los grupos en la agrupación jerárquica . En complete linkage, la distancia entre dos conglomerados se calcula como la distancia entre los dos elementos más lejos en los dos grupos.*

getDistance() y retorna un objeto del tipo lista con tres elementos:

– Matriz distancia entre las dos plantillas

– Matriz de las mejores asignaciones (distancias inter-clústers mínimas)

– Distancia total entre las dos plantillas

El proceso global consiste en calcular la distancia entre cada una de las plantillas del directorio tratado. A cada plantilla se le asigna un identificador numérico correlativo. El método principal se llama *clsHcFiles().* El resultado final es un objeto del tipo lista con los siguientes elementos:

- *distances*: matriz de distancias entre cada una de las plantillas del directorio tratado.

- fileColors: lista que contiene cada una de las agrupaciones correspondientes a cada una de las plantillas del directorio.

- *filesAsigDist*: lista que contiene las mejores asignaciones entre cada par de plantillas del directorio y la matriz de distancias entre estas plantillas.

- *equivFiles*: lista donde cada elemento es un array con los identificadores de las plantillas que son equivalentes.

- *numTiposCls*: cantidad total de diferentes números de clústers de las plantillas que se ubican en el directorio.

Una vez calculada la matriz de distancias mínimas se realiza la agrupación con el comando *hclust.*

El resultado obtenido es un objeto de la clase *hclust.* Para obtener los clústers basta con usar el comando *cutree(hcObject,numClusters).*

Para encontrar el número de clústers óptimo en los que se agrupan las plantillas se usan los índices de integridad *Dunn y Silhouette.* El rango de K's (número de clústers) para los cuales se calculan los índices de integridad viene dado por un número mínimo de clústers tal que permita agrupar plantillas con solo un clúster de diferencia y por un número máximo de clústers que es 8 (limitación funcional).

Para el cálculo de la *K* mínima primero se buscan los números de clústers de cada plantilla, y se dividen por 2. Por ejemplo, para unas plantillas analizadas se tienen plantillas de 3,4,5,6,7,8 clústers, por lo tanto, kMin= (6/2); como mínimo estas plantillas se tienen que agrupar en 3 clústers (3-4, 5-6,7-8). De este modo se evita que se agrupen clústers muy distantes entre si.

De entre las dos *K* óptimas que retorne el índice *Dunn y Silhouette* nos quedamos con la *K* más pequeña.

Para cada grupo de plantillas se busca una plantilla que llamamos *guía.* Esta plantilla es la que se usará como referencia para reasignar los identificadores de los clústers internos del resto de plantillas que pertenecen a su grupo.

Reasignando los identificadores se consigue que los diferentes clústers de cada una de las plantillas de un grupo sean lo más similares posibles.

La plantilla *guía* debe cumplir dos condiciones, que tenga el menor número de clústers internos de todas las plantillas del grupo y la otra que se la más próxima (menor distancia) respecto el resto de plantillas.

La reasignación se realiza a partir de las mejores asignaciones que se calcularon en el proceso anterior. Se buscan las mejores asignaciones de cada una de las plantillas correspondientes a un clúster respecto la plantilla *guía.*

Estos procesos de agrupamiento de plantillas y de su normalizado se realizan automáticamente para el directorio *Best* que contiene las plantillas precalculadas. Se realiza después del proceso de agrupación lanzado en el preproceso usando el comando *perl normalizeFilesR.pl*.

Esta nueva funcionalidad se integra en la página web *appl_palette.php.* Ahora en vez de listar directamente las plantillas de colores ubicadas en el directorio *Best* se muestran agrupando las plantillas por clústers e identificando las plantillas que son equivalentes.

En el proceso de agrupación y normalizado se generan una serie de ficheros donde se guarda el resultado de este proceso y se almacenan en el mismo directorio donde se ubican las plantillas analizadas.

En el fichero *hcFiles.txt* se guarda información relativa a los clústers encontrados por el algoritmo hierarchical (ver *Anexo 5*):

- Número de clústers
- Identificadores de las plantillas ordenados
- Identificadores de los clústers ordenados a partir del orden de los identificadores de las plantillas
- Asignaciones de cada plantilla a su clúster correspondiente. Las plantillas equivalentes se separan usando el carácter "," , el resto de plantillas de un mismo clúster se separan usando "-"

Para ver un resumen de la agrupación realizado sobre las mejores plantillas se habilita un icono que redirecciona al usuario a una nueva página web implementada, *appl_palette_dendogram.php.*

En esta nueva página se muestra información relativa al resultado obtenido a raíz de aplicar el hierarchical *clustering* sobre las plantillas del directorio *Best*.

Para mostrar esta información se usan los ficheros *mId.filenames.txt*, *mId_distanceFiles* y *mId_hcDendrogram.png* donde *Id* es el identificador de la microarray actual.

En el fichero *filenames.txt* se guardan los nombres de los ficheros correspondientes a cada una de las plantillas del directorio tratado ordenados por su identificador asignado en el proceso de agrupación.

En el fichero *distanceFiles.txt* se guarda una matriz correspondiente a la distancia entre cada una de las plantillas del directorio analizado.

Finalmente, el fichero *hcDendrogram.png* corresponde a la imagen del dendrograma obtenido por el *hierarchical clustering* de las plantillas analizadas.

Toda esta información se muestra en la página web implementada *appl_palette_dendogram.php*.

4.13.- Búsqueda de genes marcadores

La implementación se realiza en el fichero *gsTdStud.cc*

A partir del nivel de confianza escogido por el usuario se crean los intervalos de confianza para cada clúster :

$(\bar{x} - z * \frac{desv}{\sqrt{N}}, \bar{x} + z * \frac{desv}{\sqrt{N}})$, donde

$\bar{x}$ *es la media de las muestras del clúster ,*

N *el total de muestras del clúster ,*

z *el valor para el nivel de confianza de la distribución normal estándar ,*

desv desviación estándar del conjunto de muestras del clúster

El algoritmo implementado en C es el siguiente:

Para cada gen de la microarray

Creación Intervalos para todos los clústers

Para cada clúster

Validar si el intervalo cumple la condición respecto el basal 0 (por encima o por debajo)

Validar si el intervalo cumple con las condiciones respecto los intervalos de los otros clústers

fin para clúster

Si se cumplen todas las condiciones guardamos este gen como resultado de la búsqueda

fin para genes

Suponiendo que un intervalo genérico es (a,b) donde a>b, las condiciones de validación son las siguientes:

- Para que un clúster sea info expresado respecto el basal : a<0
- Para que un clúster sea sobre expresado respecto el basal : b>0
- Para que un clúster este sobre expresado respecto otro clúster (c,d) donde d>b : a>c y b>c
- Para que un clúster este sobre expresado respecto otro clúster (c,d) donde d>b : c>a y b>a
- Para que un clúster no se solape con otro clúster (c,d) donde d>b : (a>c y b>c) o (c>a y b>a)

Para todas las validaciones que se cumplan se mide una distancia, la suma de estas distancias se utiliza para ordenar los genes encontrados. Estas distancias se miden del siguiente modo:

- Para validación basal: si se info expresa distancia= a, si se sobre expresa distancia=b
- Para validación validación entre los dos clúster con intervalos (a,b) y (c,d) :
  - Si (a,b) se sobre expresa respecto (c,d) distancia=b-c
  - Si (a,b) se info expresa respecto (c,d) distancia=d-a
  - Si (a,b) no se solapa con (c,d) distancia= distancia entre el intervalo mayor respecto el intervalo pequeño.

Cuanto mayor sea la distancia de un gen determinado significará que los clústers condicionados están más alejados entre sí o más alejados del valor basal 0.

4.14.- Interfaz web: Integración búsqueda de genes marcadores

La integración de esta nueva funcionalidad de búsqueda de genes a partir de los intervalos de confianza se agrega en una página web existente que se llama *appl_classegs.phtml* que está disponible a través del applet PCOPGene.

Cuando el usuario lanza la búsqueda de genes se ejecuta el script de php *appl_classenewgs.php* que a su vez ejecuta la búsqueda con el script *gsTdStud* pasándole por parámetro la confianza escogida.

Los genes encontrados se guardan en un fichero para que después la página web *appl_classegsresults.phtml* los muestre por pantalla al usuario cuando finaliza la búsqueda.

Tanto los scripts como los resultados se almacenan en el directorio */var/www/cgi-bin/classes* del servidor.

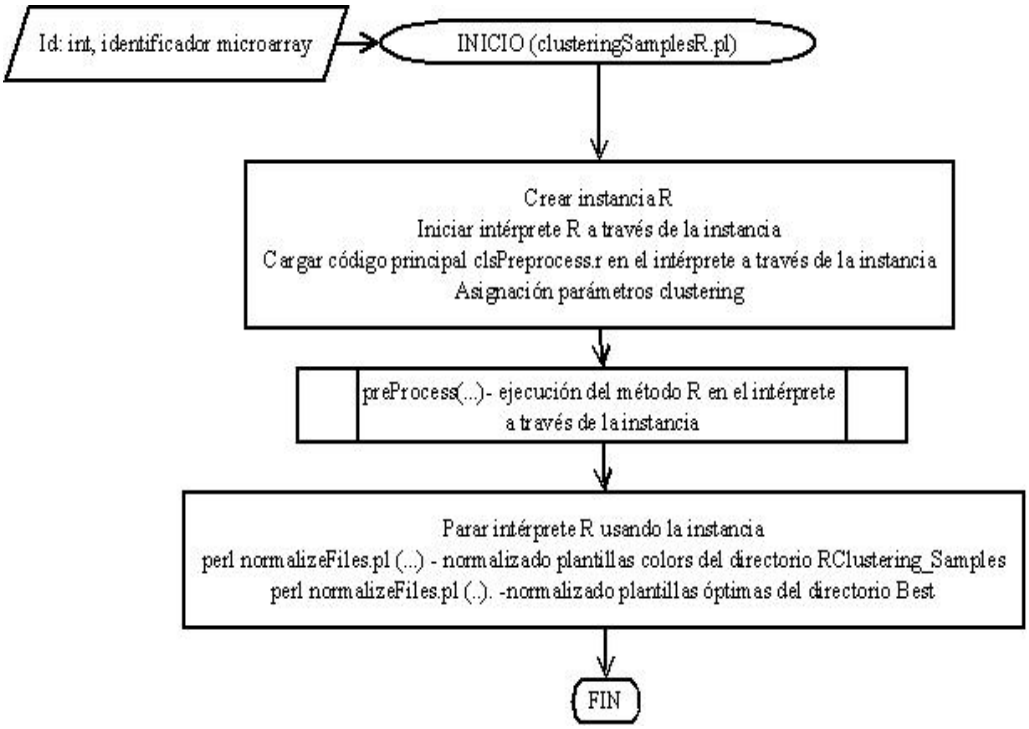
Los ficheros web se ubican en el directorio */var/www/html/applic/gexp/microarray.*

4.15..- Diagramas de flujo de los programas realizados

4.15.1.- Agrupación de condiciones muestrales y normalización de ficheros de clústers

A continuación se muestra el diagrama de flujo con los procesos más relevantes y con una descripción breve de los métodos más importantes implementados en la agrupación de condiciones muestrales.

Diagrama de flujo del script *clusteringSamplesR.pl* que se ejecuta en el preproceso:



AGRUPACIÓN DE CONDICIONES

preProcess()

Descrpción: Método main de clsPreprocess.r. Realiza la agrupación de las condiciones muestrales de una microarray determinada guardando los resultados en una serie de ficheros.

IN:

marrayNum:int, identificador microarray
marrayPath: string,path ubicación de los datos de la microarray
biclust:boolean,si es TRUE se calcula biclustering,TRUE
clMethods:string, listado de métodos de agrupación para la microarray.
clMethods=c("som","hierarchical","sota","pam")
clMethodsPC:string, listado métodos de agrupación para la microarray escalada PC.
clMethodsPC=c("som","hierarchical","sota","pam","kmeans")
clMethodsMDS:string, listado métodos de agrupación para la microarray escalada MDS.
clMethodsMDS=c("som","hierarchical","sota","pam","kmeans")
numClusters:rango int:int, números de clústers posibles para las agrupaciones. numClusters=4:16
minK:int, número mínimo de clústers para el cual se busca la mejor agrupación .minK=4
maxK:int, número máximo de clústers para el cual se busca la mejor agrupación .maxK=9
maxClusters:int,número máximo de clústers que puede tener una plantilla de colores.maxClusters=9

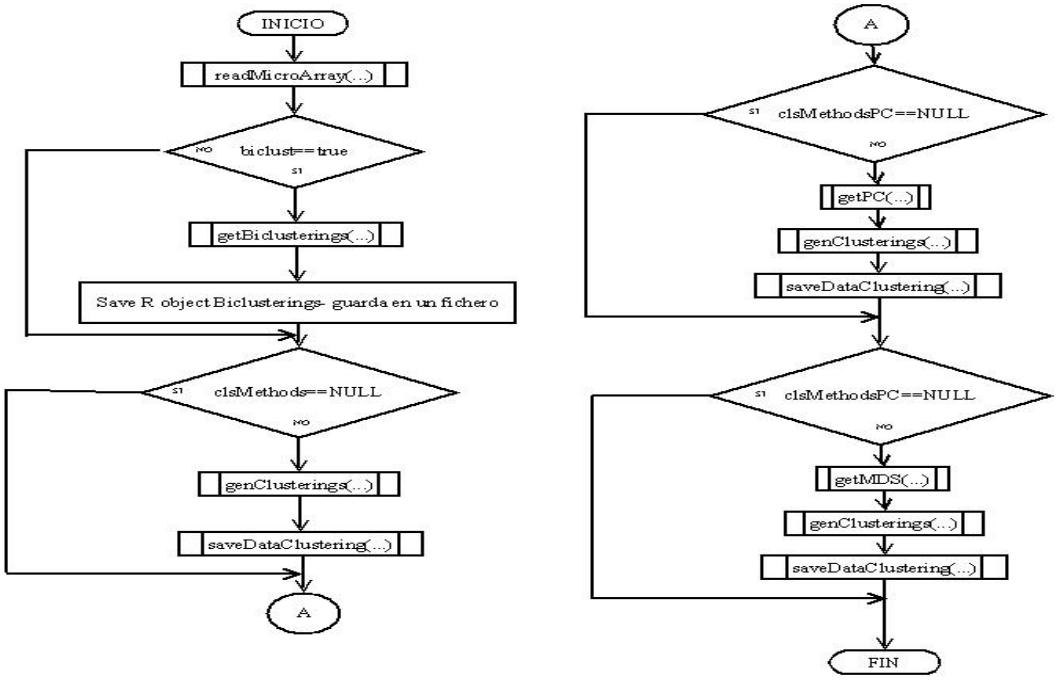
Diagrama de flujo:

createclustering()

Descripción: Método de clsPreprocess.r que se encarga de realizar las agrupaciones de la microarray dada.

IN: Los parámetros de entrada son los mismo que se le pasan al método main preProcess()

Diagrama de Flujo:



readMicroArray()

Descripción: clsReadMicro.r, realiza la lectura de los ficheros de una microarray dada
IN:
mArrayName: string,id de la microArray
mArrayPath: string, path en donde se encuentra el directorio de la microarray
OUT:
datos: list, elementos:
- micro:matriz de la microarray nGenesXn*condiciones muestrales* sin asignar nombre ni a los genes ni a las condiciones muestrales
- imputed:matriz de la microarray nGenesXnCondiciones con posibles huecos rellenados
- trans:matriz de la microarray imputed transpuesta, nCondicionesXnGenes
- microNames:matriz de la microarray condicionesXgenes con nombres asignados a los genes y a las condiciones y con los huecos rellenados

getBiclustering()

Descripción: método que calcula el biclustering bcISA, bcCC y bcBimax
IN:
data : matrix real, matriz de valores que corresponde a la microarray nCondicionesXnGenes
OUT:
list, lista de objetos que corresponde a cada uno de los biclusterings(bcIsa,bcCC y bcbimax)

genClustering()

Descripción: clsClustering.r, crea las agrupaciones de una serie determinada de métodos dados a partir de la microarray dada; usa la librería clValid para crear estas agrupaciones.
IN:
microArray:matrix real,nCondicionesXnGenes, matriz de datos a clusterizar que corresponde a la microarray
numClusters:rango int:int, números de clústers posibles para las agrupaciones
clMethods: array string, métodos de agrupación a realizar(solo métodos que soporta clValid)
OUT:
clustering: objeto de salidad de clValid, contiene agrupaciones y cálculos de integridad internos

saveDataclustering()

Descripción: clsSaveData.r,principalmente guarda los ficheros .colors resultantes de la agrupación,y los .colors con mejor indice de integridad; a parte guarda un fichero resumen por cada método de agrupación que utilizara la pagina web app_palette.php para mostrar estos resultados en la web http://revolutionresearch.uab.es[38]. También guarda los objetos mas importantes que genera R en ficheros. Genera colors y estadísticas para cada uno de los métodos de agrupación de entrada
IN:
microData: matrix nCondicionesXnGenes, matriz datos de la microarray
clustering: clValid object, objeto que genera el clValid y contiene las agrupaciones aplicadas sobre la microarray

fileDatosOrig: string, nombre raíz de los archivos sobre los que se guardarán objetos intermedios generados por R (incluye path)
fileColor: string, nombre raíz de los mejores colors según los indices de integridad (incluye path)
fileclustering: string, nombre raíz de todos los colors generados por la agrupación (incluye path)
fileEstadistica: string, nombre raíz de todos los ficheros estadísticos para cada método de agrupación (incluye path)
minK: int, k mínima para obtener la k optima
maxK: int, k máxima para obtener la k optima
maxClusters: int, numero máximo de clústers que puede tener un fichero del tipo colors

getPC()

Descripción: clsScalingData.r, reduce las dimensiones de la microarray de entrada usando las Compomentes Principales.

IN:
dataMicro, matrix real, matriz de datos de la microarray nCondicionesXnGenes

OUT:
matrix real, matriz de dos dimensiones nCondicionesX2 (componentes principales)

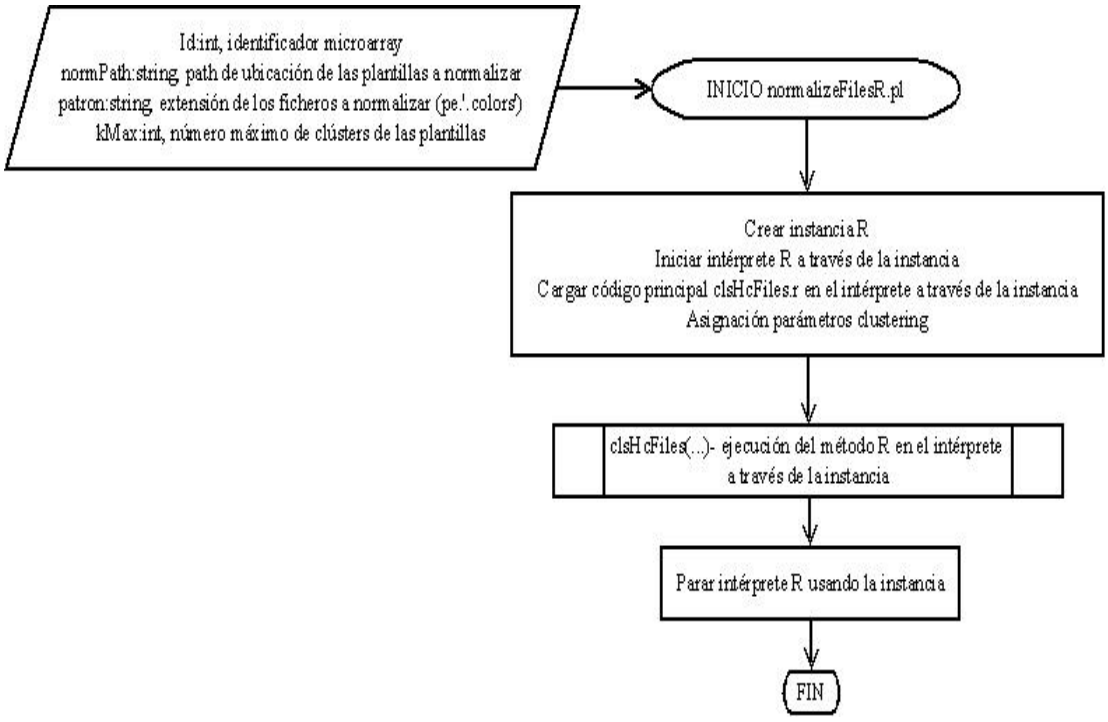
getMDS()

Descripción: clsScalingData.r, reduce las dimensiones de la microarray de entrada usando el Multi Dimensional Scaling.

IN:
dataMicro, matrix real, matriz de datos de la microarray nCondicionesXnGenes
maxDimension: int, numero máximo de dimensiones
kIdeal: int, numero de dimensión ideal
propMDS: real, porcentaje ideal de representación de componentes de la microarray

OUT:
matrix real, matriz nCondiciones*k (k dimensiones ideales)

NORMALIZADO FICHEROS DE CLÚSTERS

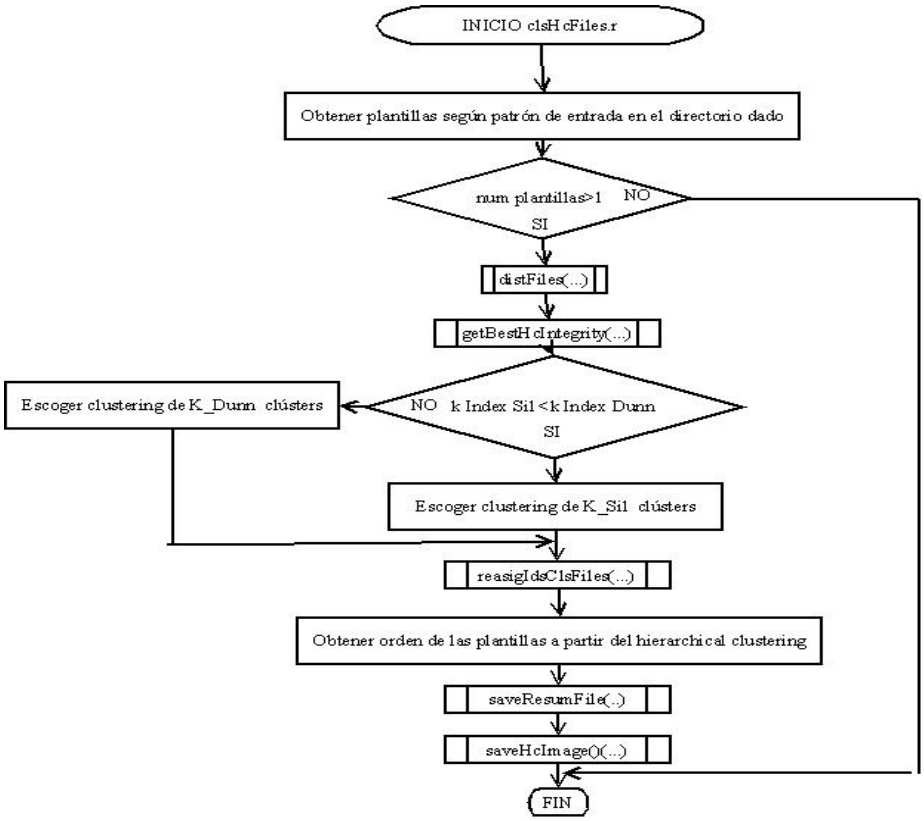


clsHcFiles()

Descripción: clsHcFiles.r, agrupa las plantillas usando el algoritmo hierarchical clustering a partir de la matriz de distancias entre cada plantilla del path. Se escoge la mejor plantilla(la más próxima al resto) para cada agrupación. A cada plantilla de estas agrupaciones se les reasigna el identificador de sus clústers teniendo en cuenta la plantilla guía correspondiente.

IN:
idMicro: int, id de la micro a la cual pertenecen los datos
path:string, path en el cual se ubican los ficheros tipo colors a analizar
patron:string, patrón que se usara para filtrar los ficheros dentro del path, por defecto ".colors", es decir, todos los ficheros colors

Diagrama de flujo:



distFiles()

Descripción: calcula la matriz de distancias entre cada cluster y a partir de esta matriz realiza otras operaciones.

IN:

fileList:array string, array de los nombres de los ficheros .colors a tratar (nombre con path)

OUT:

list, lista que contiene los siguientes elementos:
distanceFiles: matriz triangular nFilesX(nFiles-1) donde cada elemento corresponde a la distancia mínima entre cada par de ficheros
fileColors: list, cada elemento corresponde a la matriz colors de cada fichero (matriz 2Xn*condiciones muestrales*, donde la columna 1 es el idCluster y la columna 2 es el idSample)
filesAsigDist: list, para cada par de ficheros tendremos 2 elementos en la lista, uno sera una matriz con las asignaciones de clústers y el otro sera una matriz con las distancias mínimas entre cada clúster de estas asignaciones
equivFiles:list, cada elemento corresponde a un array de ids de ficheros con distancia 0 entre ellos, es decir equivalentes
numTiposCls:int,cantidad de numero de clústers diferentes que hay en todos los ficheros analizados

getBestHCIntegrity()

Descripcion: calcula el mejor k para los valores de los indices de integridad Dunn y Silhouette a partir del hierarchical clustering de un conjunto de plantillas

In:

matrDist: matrix int nFilesXnFiles, matriz de distancias entre cada par de ficheros
numTipoCls: int, cada colors tienen un numero de clústers, el total de números de clústers diferentes corresponde al valor de esta variable
kMaxFiles: int, 8 por defecto, define la k máxima de poda, es decir el numero máximo de clústers de los ficheros

Out:

list:
clustering: hierarchical clustering object, agrupación de los ficheros a partir de su matriz de distancias
kSil: int,k optimo según indice silhouette
valSil: real, valor indice silhouette
clsSil: array int, id clúster para cada fichero con kSil numero de clústers
kDunn: int,k optimo segun indice dunn
valDunn: real, valor indice dunn
clsDunn: array int, id clúster para cada fichero con kDunn numero de clústers

reasigIdsClsFiles)

Descripción: dado unos clústers de ficheros buscamos el fichero "guía" para cada clúster, estos ficheros "guía" son los que se usaran para realizar las reasignaciones de ids de clúster de *condiciones muestrales* correspondientes a cada fichero dentro del cluster correspondiente. Guarda las reasignaciones realizadas sobrescribiendo los .colors. Modifica físicamente los ficheros

IN:

clsFiles: array int, cada posición del array corresponde al id de un fichero y contiene el id clúster al que pertenece después de realizar el hierarchical clustering.
fileList: array string, array que contiene el nombre completo de los ficheros, incluye path
fileColors: list, cada elemento corresponde a una matriz tipo colors (2xnCondiciones) de cada uno de los ficheros de fileList
filesAsigDist: list, lista que contiene las matrices de asignaciones y distancias entre cada par de ficheros de fileList

saveResumFile()

Descripción: guarda una serie de ficheros resumen de todas las operaciones realizadas.

IN:

clsFiles: array int, contiene el id de los clústers de ficheros
distanceFiles: matriz de distancias entre cada par de ficheros
equivFiles: list, cada elemento corresponde a un array de ids de ficheros equivalentes
sortFiles: array int, ids de los ficheros ordenados a partir del resultado del hierarchical clustering
fileList: arrray string, nombre completo de los archivos incluido path
path : string, nombre del path donde se guardaran los ficheros
outFile: sitring, raíz principal para todos los nombre de los fichero de salida

saveHcImage()

Descripcion: guarda una imagen png del dendrograma del hierarchical clustering realizado

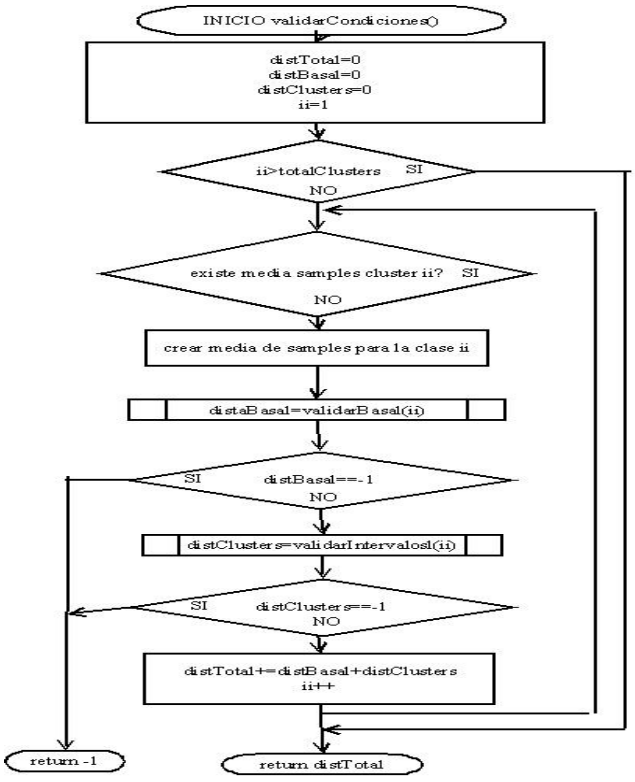
IN:

hcObj: hierarchical clustering object, objeto que contiene el hierarchical clustering realizado sobre el conjunto de ficheros .colors
path: string, path donde se guardara la imagen
outFile: string, nombre del fichero de salida

4.15.2.- Búsqueda genes marcadores

El código implementado de *gsTdeStud.cc* se genera a partir del código ya existente *gsearch.cc*. Se aprovecha la adquisición de datos y se varía el tipo de búsqueda para que use los intervalos de confianza.

Se añade el método *validarCondiciones()* que se ejecuta para cada gen de la microarray. En en caso de que la distancia total de un gen sea mayor que -1 lo guardamos en la lista de resultados.



validarBasal()

Descripción:valida las condiciones respecto el valor basal para el clúster dado como entrada usando el intervalo de la clase que se crea en el caso de que no exista ya.
IN:
clase: int, identificador de la clase a validar
OUT:
distancia: float,Distancia del clúster respecto el valor basal
-1 si no cumple la condición

validarIntervalos()

Descripción:valida las condiciones entre la clase de entrada y el resto de clústers. Para ello usa los intervalos de confianza de cada clase que se crean en el caso de que no existan ya.
IN:
clase: int, identificador de la clase a validar con el resto de clases.
OUT:
distTotal: float, suma de distancias encontradas entre el cluster de entrada y el resto de clústers validados. Si alguna de las condiciones no se cumple retorna -1

5.- Conclusiones

Los objetivos marcados se han alcanzado con creces y se han desarrollado algunas funcionalidades extra. Esto nos permite ofrecer una herramienta muy útil y versátil a los investigadores.

Es altamente útil porque mediante la agrupación de las condiciones muestrales de una microarray en clústers el investigador puede analizar el comportamiento de los genes (como se expresan) en estos estados celulares(representados por cada clúster obtenido). Además, con la búsqueda de genes marcadores para diferentes clústers pueden encontrarse genes que se sobre expresen en un determinado estado celular pero no en otro, lo que podría ser indicativo de la implicación del gen en el paso de un estado sano a uno patológico, o incluso en su curación.

Es altamente práctica porque se les facilita el trabajo al no necesitar realizar ellos mismos la agrupación de condiciones muestrales y la búsqueda de genes a partir de esta agrupación. El investigador solo tiene que seleccionar una de las agrupaciones precalculadas y a partir de ella realizar todos sus análisis y estudios. También puede manipular dichas agrupaciones precalculadas y guardarlas en su histórico personal. Con dicho fin se ha dotado al aplicativo desarrollado de una alta operativilidad, usabilidad y entendibilidad.

Por lo tanto, los dos objetivos principales, la integración de los métodos de agrupaciones de condiciones muestrales clásicos en el aplicativo web y la herramienta búsqueda de genes marcadores para estos clústers, se han cumplido con creces solucionando todos los problemas encontrados por el camino.

A nivel teórico, una de las principales conclusiones que pueden extraerse es que los actuales índices de integridad que miden cuan óptimo es una agrupación determinada no son nada precisos, al menos para en el análisis de microarrays. Si es cierto que ayudan a discriminar agrupaciones pero no se puede llegar a decir que la agrupación encontrada por el índice de integridad sea la óptima.

Uno de los aspectos más positivos del desarrollo de este proyecto ha sido ver como se pueden aplicar una serie de fundamento teóricos, estadísticos y matemáticos al mundo real, cosa que cuando estas en la carrera a veces se es un poco escéptico.

Finalmente decir que me enorgullece haber podido aportar mi granito de arena a un proyecto conjunto que puede llegar a descubrir , por ejemplo, los genes responsables de diferentes tipos de cáncer.

6.- Bibliografía

[1]http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview : paquete estadístico CLUTO
[2]http://www.r-project.org/ : paquete estadístico R-STATISTICS
[3]http://cran.r-project.org/ : repositorio de paquetes estadísiticos para R-STADISTICS
[4]http://cran.es.r-project.org/web/packages/pcaPP/index.html : paquete para cálculo PCA
[5]http://cran.es.r-project.org/web/packages/som/index.html : paquete para realizar agrupación SOM
[6]http://cran.es.r-project.org/web/packages/smacof/index.html : paquete para realizar el escalado MDS
[7]http://stat.ethz.ch/R-manual/R-devel/library/stats/html/cmdscale.html : método de R-STATISTICS para el calculo del escalado MDS
[8]http://cran.es.r-project.org/web/packages/biclust/index.html : paquete para realizar varios tipos de bliclustering
[9]http://cran.es.r-project.org/web/packages/clusterSim/index.html : paquete que calcula varios índices de integridad
[10]http://cran.r-project.org/web/packages/isa2/index.html : paquete que realiza el biclustering del tipo ISA
[11] http://cran.r-project.org/web/packages/clValid/index.html : paquete que realiza varios tipos de agrupación e índices de integridad
[12]http://cran.r-project.org/web/packages/impute/index.html : paquete para rellenado de huecos en la matriz de datos
[13] http://cran.r-project.org/web/packages/gtools/index.html : paquete que permite realizar cálculos de combinatoria
[14]Huerta M, Cedano J, Peña D, Rodriguez A, Querol E. (2009) PCOPGene-Net: holistic characterisation of cellular states from microarray data based on continuous and non-continuous analysis of gene-expression relationships. *BMC Bioinformatics.*, 9;10:138 .
[15] Delicado, P. and Huerta, M. (2003): *'Principal Curves of Oriented Points: Theoretical and computational improvements'*. Computational Statistics 18, 293-315.
[16] http://es.wikipedia.org/wiki/Chip_de_ADN : Definición microarray
[17] http://www.ebi.ac.uk/microarray-as/ae/ : Data Base microarrays
[18] http://www.ncbi.nlm.nih.gov/sites/entrez?db=geo&term=q6 : Data Base microarrays
[19] http://revolutionresearch.uab.es/downloads/webmicroarray/cluster1_3.html#inici : Microarray clustering
[20] Beatriz Pontes.*"Técnicas de Evaluación en Algoritmos de Biclustering sobre Datos de Expresión Genómica"*, http://www.lsi.us.es/docs/doctorado/memorias/Pontes,%20Beatriz.pdf
[21] P. Sneath and R. Sokal. *"hierarchical clustering"*, 1973.
[22] J. B. MacQueen (1967). *"Some Methods for classification and Analysis of Multivariate Observations*, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability"*, Berkeley, University of California Press, 1:281-297
[23] http://en.wikipedia.org/wiki/K-medoids : Definición K-medoids (PAM) y ejemplo algoritmo.
[24] T. Kohonen. *"Self–Organizing Maps"*. Spring Verlag, Berlin, 1997.
[25] L. Yin, C.-H. Huang, and J. Ni. "*clustering of gene expression data: performance and similarity analysis"* . BMC Bionformatics, 7:doi: 10.1186/1471–2105–7–S4–S19, 2006
[26] Calinski, R. B. and Harabasz, J. (1974). *"A denrite method for cluster analysis. Communications in Statistics"* 3, 1–27
[27] Dunn JC (1974). *"Well Separated Clusters and Fuzzy Partitions."* Journal on Cybernetics, 4,95-104.
[28] Hartigan, J. A. (1975). *"clustering Algorithms"*.
[29] Rousseeuw PJ (1987). *"Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis."* Journal of Computational and Applied Mathematics, 20, 53-65.
[30] Handl J, Knowles J, Kell DB (2005). *"Computational Cluster Validation in Post-Genomic Data Analysis."* Bioinformatics, 21(15), 3201-12.
[31] http://es.wikipedia.org/wiki/Intervalo_de_confianza : Definición intervalos confianza.
[32] http://es.wikipedia.org/wiki/Distribuci%C3%B3n_t_de_Student : Definición t de Student
[33] http://www.omegahat.org/RSPerl/ : librería R para la implementación de operaciones R en Perl
[34] http://search.cpan.org/~gmpassos/Statistics-R-0.02/lib/Statistics/R.pm : librería R para la implementación de operaciones R en Perl
[35]Scherf U, Ross DT, Waltham M, Smith LH, Lee JK, Tanabe L, Kohn KW, Reinhold WC, Myers TG, Andrews DT, Scudiero DA, Eisen MB, Sausville EA, Pommier Y, Botstein D, Brown PO, Weinstein JN.. 2000. A gene expression database for the molecular pharmacology of cancer. *Nature Genetics* 24(3):236-44.
[36]Cedano J, Huerta M, Querol E. (2008) NCR-PCOPGene: An Exploratory Tool for Analysis of Sample-Classes Effect on Gene-Expression Relationships *Advances in Bioinformatics*, vol. 2008
[37]Huerta M, Cedano J, Querol E. (2008) Analysis of nonlinear relations between expression profiles by the principal curves of oriented-points approach. *J Bioinform Comput Biol*. 6:367-386.
[38]http://revolutionresearch.uab.es : Web server for on line microarray analysis supported by the Institute of Biotechnology and Biomedicine of the Autonomous University of Barcelona (IBB-UAB).

[39]Cedano J, Huerta M, Estrada I, Ballllosera F, Conchillo O, Delicado P, Querol E. (2007) *A web server for automatic analysis and extraction of relevant biological knowledge*.*Comput Biol Med.*37:1672-1675.

7.- Anexos

Anexo 1 - Resultados integridad de la agrupación librería *som*

Resultados obtenidos al aplicar el algoritmo som de la librería de R *som* a la microarray 17

CALCULO INTEGRIDADES SOM					
xDim/yDim	Num. Clust.	Calinski-Harabasz	xDim/yDim	Num. Clust.	Hartigan Index
1/2	2	45.29525	1/3	3	9.00101
1/3	3	28.60326	1/4	4	2.954459
1/4	4	20.35231	1/5	5	3.190721
2/2	4	30.83795	1/5	6	-14.81250
1/5	5	16.33394	1/6	7	0.9610587
1/6	6	13.24954	1/7	7	1.592263
2/3	6	23.37993	1/7	7	-23.76523
1/7	7	11.35757	1/8	8	0.9146093
1/8	8	9.854467	1/9	9	2.748154
2/4	8	17.03204	1/9	9	-24.32977

DATOS EXTRAÍDOS DEL OBJETO RESULTANTE DE APLICAR EL ALGORITMO SOM A LA MICROARRAY 17										
xDim/yDim	Id Cluster	X	Y	nObj	qError	xDim/yDim	Id Cluster	X	Y	nObj	qError
1-2	1	0	0	47	66.50952						
	2	0	1	71							
1-3	1	0	0	38	66.1554						
	2	0	1	19							
	3	0	2	61							
1-4	1	0	0	36	63.41123	2-2	1	0	0	28	106.0781
	2	0	1	11			2	1	0	33	
	3	0	2	18			3	0	1	19	
	4	0	3	53			4	1	1	38	
1-5	1	0	0	34	61.66456						
	2	0	1	6							
	3	0	2	14							
	4	0	3	13							
	5	0	4	51							
1-6	1	0	0	33	60.2772	2-3	1	0	0	23	106.9266
	2	0	1	5			2	1	0	15	
	3	0	2	9			3	0	1	9	
	4	0	3	10			4	1	1	9	
	5	0	4	11			5	0	2	29	
	6	0	5	50			6	1	2	33	
1-7	1	0	0	32	59.70014						
	2	0	1	6							
	3	0	2	2							
	4	0	3	11							
	5	0	4	8							
	6	0	5	11							
	7	0	6	48							
1-8	1	0	1	31	59.29405	2-4	1	0	0	23	102.0151
	2	0	2	5			2	1	0	15	
	3	0	3	3			3	0	1	5	
	4	0	4	8			4	1	1	4	
	5	0	5	8			5	0	2	4	
	6	0	6	9			6	1	2	6	
	7	0	7	7			7	0	3	29	
	8	0	8	47			8	1	3	32	

Anexo 2 - Ejemplo implementación MDS y PC

Ejemplo escalado MDS:

```
< matriz
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
```

```
[1,]  0.107 -0.08 -0.09  0.05  0.01  0.12 -0.20 -0.09
[2,]  0.020 -0.23 -0.21 -0.03 -0.01 -0.32 -0.19 -0.14
[3,]  0.030 -0.11 -0.11  0.08 -0.02 -0.31 -0.14 -0.18
[4,] -0.120 -0.28 -0.29 -0.05 -0.12 -0.25 -0.08 -0.11
[5,] -0.460 -0.39 -0.37 -0.38 -0.31 -0.28 -0.16 -0.20
[6,]  0.060 -0.07 -0.06  0.02 -0.28  0.01 -0.41 -0.12
[7,] -0.220 -0.43 -0.37 -0.33 -0.31 -0.11 -0.21 -0.38
[8,] -0.180 -0.41 -0.38 -0.24 -0.28 -0.15 -0.30 -0.22
      < distMatr= dist(matriz)
    1     2     3     4     5     6     7
2 0.4974626
3 0.4534633 0.2022375
4 0.5568923 0.2426932 0.3504283
5 0.9782581 0.7072482 0.8267406 0.5426785
6 0.3811942 0.5317894 0.5050743 0.5800862 0.8847599
7 0.8317025 0.6366318 0.7547185 0.5126402 0.3542598 0.7412827
8 0.7371357 0.5154610 0.6555913 0.4009988 0.3691883 0.6252200 0.2154066
      < mdsMatriz= cmdscale(distMatr,k=6,eig=TRUE)
$points
             [,1]      [,2]       [,3]       [,4]       [,5]       [,6]
[1,] -0.43005642  0.12910393  0.166902791 -0.06655176 -0.003330968  0.022622459
[2,] -0.14545453 -0.20136374 -0.019730847  0.07237183  0.040477699  0.053706750
[3,] -0.28217251 -0.21215046 -0.060532107  0.02838603 -0.076680499 -0.010028626
[4,]  0.00911523 -0.16865616  0.040057374 -0.04635506  0.049514642 -0.073371401
[5,]  0.50617250 -0.04549211 -0.042314746 -0.14228331 -0.023054647  0.031349230
[6,] -0.29965057  0.27487261 -0.154399285 -0.01756963  0.002459757 -0.013457658
[7,]  0.37453581  0.12577653  0.073669855  0.10353557 -0.075365069 -0.015194813
[8,]  0.26751049  0.09790940 -0.003653035  0.06846633  0.085979085  0.004374058

$eig
[1] 0.84364996 0.23369837 0.06458485 0.04858186 0.02359100 0.01029402
      > sum(cumsum(mdsMatriz$eig)/sum(mdsMatriz$eig)<=0.95)
[1] 3
      > sum(mdsMatriz$eig>0)
[1] 6
```

Por lo tanto la dimensión mínima que mejor resume toda la información de las variables es k=3. Para obtener los nuevos valores de las variables para cada condición usamos el atributo *points* que retorna el método *cmdscale*: mdsMatriz$points[,1:k]

```
      > mdsMatriz$points[,1:3];
            [,1]        [,2]         [,3]
[1,] -0.43005642  0.12910393  0.166902791
[2,] -0.14545453 -0.20136374 -0.019730847
[3,] -0.28217251 -0.21215046 -0.060532107
[4,]  0.00911523 -0.16865616  0.040057374
[5,]  0.50617250 -0.04549211 -0.042314746
[6,] -0.29965057  0.27487261 -0.154399285
[7,]  0.37453581  0.12577653  0.073669855
[8,]  0.26751049  0.09790940 -0.003653035
```

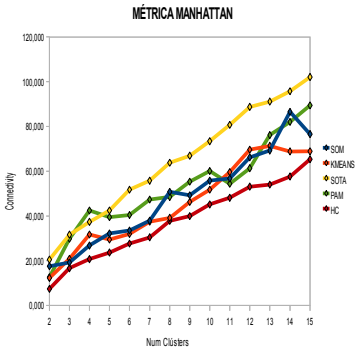
Ejemplo escalado PC:

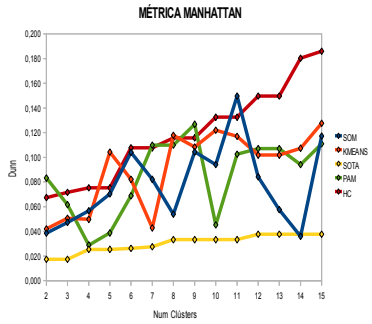
```
      > matriz
    [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
[1,]  0.107 -0.08 -0.09  0.05  0.01  0.12
[2,]  0.020 -0.23 -0.21 -0.03 -0.01 -0.32
[3,]  0.030 -0.11 -0.11  0.08 -0.02 -0.31
[4,] -0.120 -0.28 -0.29 -0.05 -0.12 -0.25
[5,] -0.460 -0.39 -0.37 -0.38 -0.31 -0.28
```

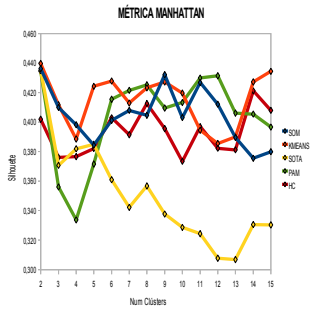
```
[6,]  0.060 -0.07 -0.06  0.02 -0.28  0.01
        > pcaMatr=PCAgrid(matriz)
        > proPCA$scores
          Comp.1      Comp.2
1  0.33756896  0.13307662
2 -0.08478308  0.07732510
3  0.04514911  0.13308273
4 -0.13659333 -0.08664403
5 -0.37826703 -0.44993529
6  0.33754209 -0.13010486
```

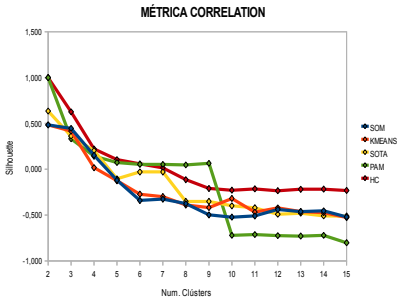
Anexo 3 - Evolución índices de integridad según métrica

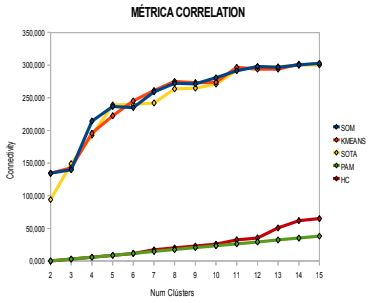
Gráficas de evolución de los índices de integridad silhouette, dunn y connectivity según métrica manhattan y correlation para todos los algoritmos de agrupación implementados aplicados sobre la microarray m17 escalada con PC.











Anexo 4 - Ejemplo fichero estadística

A continuación se muestra el contenido del fichero 17_estadistica_mds_som.txt:

```
;INTEGRIDADES:

 K Real K Nueva Fusiones Outlayers ClsSin Samples Calinsky Hartigan      Dunn
      4        4         0         0              0         0        0 0.1342878
      5        5         0         0              0         0        0 0.1167195
      6        6         0         0              0         0        0 0.1481313
      7        7         0         0              0         0        0 0.1337351
      8        8         0         0              0         0        0 0.1337351
      9        8         1         0              0         0        0 0.1473378
     10        9         1         0              0         0        0 0.1565508
     11       10         1         0              0         0        0 0.1805253
     12       11         1         0              0         0        0 0.1858766
     13       11         2         0              0         0        0 0.1214188
     14       13         1         0              0         0        0 0.1922897
     15       12         3         0              0         0        0 0.1071973
```

```
     16      13        3         0             0        0        0 0.1106986
 Silhouette
  0.3350244
  0.3338487
  0.3000921
  0.2775725
  0.2850829
  0.2977357
  0.2674198
  0.2777863
  0.2798427
  0.2644076
  0.2643060
  0.2678738
  0.2603435
[BEST]
;MEJOR K PARA DUNN
dunn.k= 10
;Num Clusters
dunn.numClusters= 9
;Valor Dunn
dunn.val= 0.157
;Valor Silhouette
dunn.val.silhouette= 0.267
;MEJOR K PARA SILHOUETTE
silhouette.k= 4
;Num Clusters
silhouette.numClusters= 4
;Valor Silhouette
silhouette.val= 0.335
;Valor Dunn
silhouette.val.dunn= 0.134
------------------

;OUTLAYERS:

;Numero minimo de condiciones muestrales para cada cluster: 6
```

Anexo 5 - Ejemplo fichero de agrupación de ficheros .colors

Contenido del fichero 17_hcFiles.txt correspondiente la agrupación de las plantillas del directorio *Best* encontradas para la microarray 17:

```
[CLUSTERS]
numClusters=4
cls=1,4,3,2
idsOrdenados=12,28,10,16,14,30,1,2,3,5,6,26,8,19,22,24,25,18,17,20,21,23,29,31,7,27,15,9,11,4,13
1.files=17_mds_som_silhouette_4.colors-17_som_silhouette_4.colors-17_mds_pam_silhouette_4.colors-
17_pam_silhouette_4.colors-17_mds_sota_silhouette_5.colors-17_sota_silhouette_5.colors-
17_hierarchical_descarte_dunn_13.colors-17_hierarchical_descarte_silhouette_7.colors-
17_hierarchical_dunn_7.colors-17_mds_hierarchical_descarte_silhouette_7.colors-
17_mds_hierarchical_dunn_6.colors-17_pc_sota_silhouette_4.colors-17_mds_kmeans_silhouette_5.colors-
17_pc_hierarchical_silhouette_6.colors-17_pc_pam_silhouette_5.colors-17_pc_som_silhouette_5.colors
4.files=17_pc_sota_dunn_9.colors-17_pc_hierarchical_descarte_silhouette_11.colors-
17_pc_hierarchical_dunn_8.colors-17_pc_kmeans_dunn_8.colors-17_pc_pam_dunn_9.colors-
17_pc_som_dunn_9.colors
3.files=17_sota_descarte_dunn_12.colors-17_sota_dunn_11.colors-17_mds_kmeans_dunn_7.colors-
17_som_dunn_9.colors
2.files=17_pam_dunn_9.colors-17_mds_pam_dunn_7.colors-17_mds_som_dunn_10.colors-
17_mds_hierarchical_descarte_dunn_15.colors-17_mds_sota_dunn_7.colors
```

8.- Resumen

El trabajo realizado se divide en dos bloques bien diferenciados, ambos relacionados con el análisis de microarrays. El primer bloque consiste en agrupar las condiciones muestrales de todos los genes en grupos o clústers. Estas agrupaciones se obtienen al aplicar directamente sobre la microarray los siguientes algoritmos de agrupación: SOM,PAM,SOTA,HC y al aplicar sobre la microarray escalada con PC y MDS los siguientes algoritmos: SOM,PAM,SOTA,HC y K-MEANS. El segundo bloque consiste en realizar una búsqueda de genes basada en los intervalos de confianza de cada clúster de la agrupación activa. Las condiciones de búsqueda ajustadas por el usuario se validan para cada clúster respecto el valor basal 0 y respecto el resto de clústers , para estas validaciones se usan los intervalos de confianza. Estos dos bloques se integran en una aplicación web ya existente , el applet PCOPGene, alojada en el servidor: http://revolutionresearch.uab.es.

El treball realitzat es divideix en dos blocs ben diferenciats, el dos relacionats amb l´anàlisis de microarrays. El primer bloc consisteix en agrupar les condicions experimentals de tots el gens en grups o clústers. Aquestes agrupacions s'obtenen al aplicar directament sobre la microarray els següents algoritmes d'agrupació: SOM,PAM, SOTA,HC i a l'aplicar sobre la microarray escalada amb PC y MDS els següents algoritmes d'agrupació: SOM,SOTA,PAM,HC i K-MEANS. El següent bloc consisteix en realiztar una cerca de gens basada en els intèrvals de confiança de cada clúster de l'agrupació activa. Les condicions de cerca ajustades per l'usuari es validen per cada classe respecte el valor basal 0 i respecte el reste de classes, per aquestes validacions s'utilitzen els intèrvals de confiança. Aquests dos blocs s'integren en un aplicatiu web ya existent , l'applet PCOPGene, allotjada al servidor: http://revolutionresearch.uab.es.

The work is divided into two distinct, both related to the analyze of microarrays. The first block is to group the experimental conditions of all genes in groups or clusters. These clusters are obtained by applying directly on the microarray the following algorithms: SOM, PAM, SOTA, HC and applying microarray climbing on PC and MDS following algorithms: SOM, PAM, SOTA, HC and K-MEANS. The second block is to perform a search for genes based on confidence intervals for each cluster from the active agrupation. Search conditions set by user are validated for each cluster about the vasal value 0 and about the other clusters, these validations are used confidence intervals. These two blocks are integrated into an existing web application, the applet PCOPGene, available in the web server: http://revolutionresearch.uab.es.