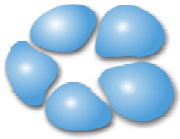
ANÁLISIS DE DATOS DE MICROARRAYS

Abril 2010

José González Maestre
jogonmae@inf.upv.es



ibime
Informática Biomédica

UNIVERSIDAD POLITECNICA DE VALENCIA

AGRADECIMIENTOS:

Gracias a Alfredo y Juanmi por sus buenos consejos y su ayuda.

ÍNDICE

Resumen

Desde que en 1956, Rosalind Franklin, Watson y Crick descubrieran a través de las imágenes de rayos-X el modelo de la estructura del DNA, los avances tecnológicos y biológicos buscan entender los mecanismos biológicos en los que el código genético juega un papel tan importante.

La tecnología de microarrays se ha convertido en una herramienta esencial en el descubrimiento de la información genética. Específicamente, la función de los microarrays de expresión genética es detectar el comportamiento de genes determinados bajo condiciones específicas. A partir de ahí, se abre un abanico de posibilidades dentro de la epidemiología, la farmacología o cualquier rama de la medicina en general. En nuestro caso la oncología.

La experimentación basada en microarrays está sujeta a un pre-proceso de la información obtenida, debido en gran medida al proceso de construcción, hibridación y segmentación del propio microarray.

Con este proyecto se revisa la fabricación de los microarrays, el proceso de hibridación, el escaneado y la normalización de los datos de expresión genética. Como objetivo práctico del mismo se estudian mediante experimentos computacionales la clasificación automática de casos de tumores cerebrales biopsiados.

Palabras Clave

Clasificación automática, Bioinformática, tumor cerebral, Affymetrix, microarray, genechip, biochip, normalización.

Introducción

El creciente desarrollo de la ciencia trae consigo la aparición de nuevas disciplinas, y muchas de ellas provenientes de la fusión de algunas ciencias ya existentes. Estas nuevas áreas de estudio hacen uso de equipos sumamente sofisticados, los cuales producen, a su vez, nuevas estructuras de datos.

En ese contexto, una disciplina relativamente nueva, y que ha logrado un amplio desarrollo en la actualidad es la bioinformática.

La Bioinformática, se dedica a la investigación y desarrollo de herramientas útiles para entender el flujo de información desde los genes hasta sus estructuras moleculares, su función bioquímica, su conducta biológica y finalmente, su influencia en las enfermedades y en la salud. En nuestro caso realizaremos el análisis bioinformático de expresión genética en tumores cerebrales.

Los datos que se analizan en bioinformática provienen mayormente, de las expresiones de genes (o expresión genética), las cuales pueden llegar a ser miles en una sola observación. Existen varias maneras de medir la expresión genética, una de ellas, es la tecnología de los microarrays. Ésta, permite analizar simultáneamente miles de genes; sin embargo, el costo por observación es muy alto.

A su vez, nos vamos a apoyar en la estadística, para posteriormente tomar decisiones sobre los resultados obtenidos. Por esa razón, constantemente se proponen metodologías para analizar estructuras de información emergentes.

El desarrollo del presente trabajo se justifica en el marco del estudio de expresiones genéticas mediante estas nuevas estructuras de datos.

En efecto, un tipo de estructura de datos relativamente nuevo es el proveniente de la aplicación de la tecnología de microarrays. Este tipo de estructura de datos conocido comúnmente como microarray data, consiste en un gran número de moléculas de ADN ordenadas sobre un sustrato sólido de manera que formen una matriz de secuencias en dos dimensiones. Estos fragmentos de material genético pueden ser secuencias cortas, llamadas oligonucleótidos, o de mayor tamaño estabilizado en forma de cDNA (ADN complementario).

La experimentación con micorarrays presenta como característica principal que el número de variables (genes) es considerablemente mayor en comparación a la cantidad de observaciones analizadas.

En este proyecto, la expresión genética está asociada a la prolifereación de tumores cerebrales; es decir, cada conjunto de genes provenientes de tejido biológico de un individuo relacionado a un tipo de tumore cerebral. Más aún, en las investigaciones biomédicas actuales, se utiliza para descubrir subtipos tumorales. Desde el punto de vista estadístico, se puede establecer que la medición de los genes provenientes de las expresiones genéticas se puede considerar como variables predictivas; mientras que los tipos, subtipos de cáncer o ausencia de cáncer, codificados adecuadamente, pueden ser utilizados como las clases.

En el análisis de clasificación supervisada, se dispone de un conjunto de pacientes de los cuales se ha observado la expresión genética, y para los cuales se conocen las clases a las que pertenecen. El objetivo principal en este tipo de análisis es clasificar los diferentes tipos de tumores que tenemos en la base de datos y estimando a su vez el error de mala clasificación para el clasificador elegido.

Si se considera que muchas de las técnicas estadísticas tradicionales han sido diseñadas para analizar un número considerable de observaciones en comparación a la cantidad de variables en estudio; entonces cuando sucede lo contrario, como en el caso de los datos provenientes de expresiones genéticas, se pueden obtener resultados poco satisfactorios.

Si bien es cierto que en la actualidad se han desarrollado varios métodos que trabajan con datos provenientes de expresiones genéticas en clasificación supervisada; muchos de ellos presentan algoritmos complejos, lo que conlleva que el tiempo de procesamiento de la información sea alto.

Finalmente, cabe mencionar que la metodología que se expondrá se aplicó a una base de datos multicéntrica de microarrays con información de expresión genética tumoral, pero eso no impide que pueda también aplicarse a problemas que provengan de otras áreas que trabajen con datos de estructura similar a la de los microarrays.

Objetivos

Objetivo Principal

Realizar un análisis bioinformático de expresión genética en tumores cerebrales, estimando el error de clasificación mediante expresión genética de nuevos casos.

Objetivos Específicos

Como objetivos específicos del proyecto nos plantearemos cubrir un análisis completo de microarrays de expresión genética:

- Control de calidad de microarrays de expresión genética

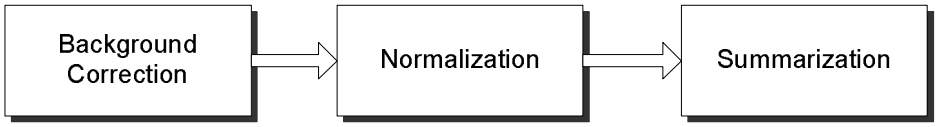
  Cada uno de los microarrays contiene un conjunto de gene probes que hacen el papel de marcadores de la muestra. El valor de la intensidad que se obtiene de ellos es utilizado para generar el control de calidad del microarray.

  - Inspección visual del slide de cada microarrays en busca de defectos de la imagen: rascones, contaminación debida al polvo o gotas de humedad, etc.…
  - Visualización las intensidades de nuestros microarrays: la observación de patrones diferentes a una degradación mayor en 5' o una velocidad de degradación muy diferente de algún microarray respecto al resto indicaría una mala calidad en la hibridación del microarray y no tendría un mínimo de calidad para ser utilizado.

- Pre procesado de microarrays

  En un análisis de comparación se toman dos muestras de biochips del mismo tipo y se contrastan las expresiones/intensidades de los genes involucrados en el estudio, para así poder detectar y cuantificar variaciones (changes) en dichas expresiones. Uno de los arrays se designa como base del estudio (baseline array) y el otro como experimento (experiment array). En el proceso de análisis se utilizan dos procesos: uno para generar un valor cualitativo del contraste entre arrays, denominado Change p-value; el otro nos sirve para hacer una estimación cuantitativa, denominado Signal Log Ratio.

  Para que los valores de expresión genética de los microarrays sean comparables entre si, es necesario eliminar el efecto de los errores sistemáticos acumulados durante la obtención del microarray. El pre proceso general de los datos consta de 3 pasos básicos:



- Corrección del Background

  Sabemos que existe una cantidad de hibridación no específica contenida en el background y que afecta a la sensibilidad y especificidad del chip. Este paso del

pre proceso permite minimizar los efectos que dicha hibridación causa sobre la señal de cada probe cell.

- Normalización

  Es necesario aplicar un escalado y/o normalizado para poder ajustar la señal, que puede contener errores causados por factores técnicos y biológicos. La normalización recoge la información de los probe cell correspondientes a cada probe set de cada chip y los ajusta a un valor comparable, de manera que si alguno esta sobre-expresado o infra-expresado con respecto al total, regulando, así, su valor de expresión génica. Robust Multiarray Average (RMA) es una conocida técnica de normalización.

- Sumarización

  Acumula la información de los probe pertenecientes a un mismo probe set en un valor cuantitativo (signal) que representa el nivel relativo de expresión correspondiente a su transcripción genética.

- Control de calidad de datos pre procesados
  - Comparación entre laboratorios. A partir de las muestras pre procesadas de los diferentes laboratorios se puede analizar si la procedencia de los datos afecta de manera significativa a la clasificación de las muestras, debido a errores de medición o dependencias del instrumental y/o personal de cada laboratorio. En caso de existir dicha dependencia se tendrán que tomar medidas para poder utilizar conjuntamente todos los datos de los laboratorios, como por ejemplo la normalización de dichos datos.

- Selección de genes diferenciados por expresión
  - Métodos de remuestreo, p.e. random sampling: Repetimos una separamos de los casos en un 70% - 30% aleatoriamente para evaluar la capacidad de clasificación basada en la selección de genes. Se selecciona aquel conjunto de genes que obtiene una mejor clasificación. Otros métodos similares son kRSTT y validación cruzada.
  - Test estadísticos univariantes, o multivariantes: se seleccionan los genes con mayor poder discriminante mediante un contraste de hipótesis.
  - Como alternativa de la selección de genes, se puede estar interesado en extraer características de menor dimensionalidad que resuman la información del conjunto de genes de interés.

- Clasificación y evaluación de tumores cerebrales por firmas genéticas
  - Determinar la capacidad discriminante de firmas genéticas basadas en microararys de expresión genética de tumores cerebrales.
  - Determinar que preguntas médicas son más factibles para su resolución mediante firmas de expresión genética.

Biotecnología y biología

La cadena de ADN

Para entender la mecánica de un biochip debemos conocer la estructura de la cadena de ADN. Ésta se compone de dos cadenas de polímeros cuyas unidades básicas son los nucleótidos. Cada nucleótido se compone de un azúcar y una base nitrogenada. Estos nucleótidos se concatenan entre sí mediante un grupo fosfato. La unión de dos polímeros a través de sus correspondientes bases nitrogenadas conforma la estructura helicoidal del ADN.

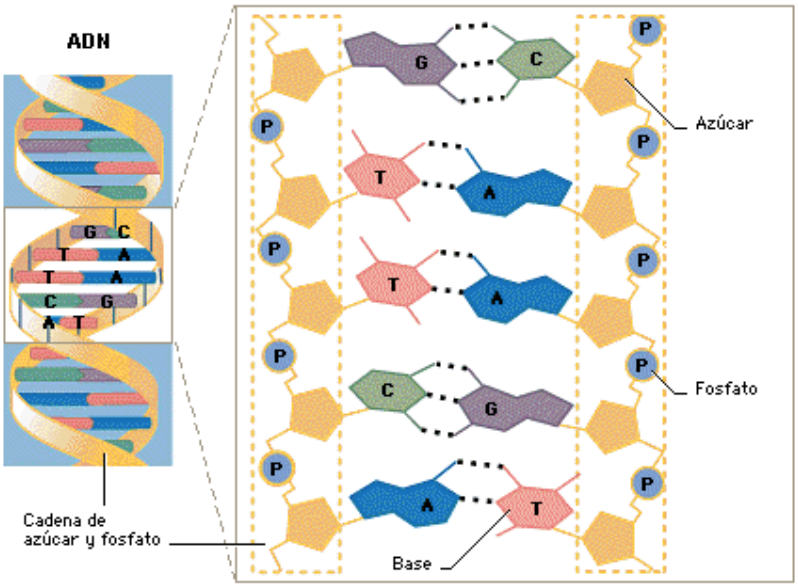


Ilustración 1. Estructura de ADN

Los azúcares de cada nucleótido son un compuesto de 5 carbonos numerados. El grupo fosfato establece la unión entre el carbono 5' de un azúcar y el 3' de otro para unir dos nucleótidos. De este modo se establece una direccionalidad dentro de la estructura de cada polímero.

Por otro lado, la unión de dos polímeros no es arbitraria. Las bases nitrogenadas son de cuatro tipos: adenina (A), citosina (C), guanina (G) y timina (T). Y existe una relación única entre estos tipos que establece que las posibles uniones entre ellos son: A-T y C-G. Por tanto la secuencia de uno de los polímeros de la cadena de ADN es única respecto a su polímero opuesto.

Microarrays

Un microarray consiste en un gran número de moléculas de ADN ordenadas sobre un sustrato sólido de manera que formen una matriz de secuencias en dos dimensiones.

Estos fragmentos de material genético pueden ser secuencias cortas llamadas oligonucleótidos, o de mayor tamaño, cDNA (ADN complementario, sintetizado a partir de mRNA), o bien productos de PCR (replicación in vitro de secuencias de ADN mediante la reacción en cadena de la Polimerasa). A estos fragmentos de ADN de una sola hebra inmovilizados en el soporte, se les denomina a menudo "sondas" (*probe sequences*).

Los ácidos nucleicos de las muestras a analizar se marcan por diversos métodos (enzimáticos, fluorescentes, etc.) y se incuban sobre el panel de sondas, permitiendo la hibridación (reconocimiento y unión entre moléculas complementarias) de secuencias homólogas.

Durante la hibridación, las muestras de material genético marcadas (*target sequences*) se unirán a sus complementarias inmovilizadas en el soporte del chip, permitiendo la identificación y cuantificación del ADN presente en la muestra (mutaciones, patógenos, etc.). Con posterioridad, el escáner y las herramientas informáticas nos permiten interpretar y analizar los datos obtenidos.

Los pasos en el diseño y fabricación de un microarray se muestran en la **Tabla 1**:

Operaciones de Diseño y Fabricación	Técnicas utilizadas
➔ Elección del tipo de ADN	Oligonucleótidos, clones cDNA, PCR
➔ Marcaje de sondas o muestras	Fluorescencia, Enzimático,…
➔ Selección del material de soporte	Vidrio, Plástico, membranas,…
➔ Inmovilización de las sondas	Activa, Pasiva, Covalente,…
➔ Hibridación y Lavado	Impresión, Síntesis in situ,…
➔ Detección de hibridación	Escáneres, fluorimetrías,…
➔ Procesamiento de datos	Software específico

Tabla 1. Diseño de microarrays

Existen dos técnicas principales en la fabricación de un biochip: la síntesis in situ y la impresión de cDNA sintetizado en el soporte.

En la **síntesis in situ** los oligonucleótidos sonda son foto-químicamente sintetizados directamente en el chip.

Affymetrix GeneChip's

Tecnología de Array de Oligonucleótidos

La fabricación de un microarray de oligonucleótidos usa la síntesis in situ, lo cual quiere decir que las secuencias de ADN sonda se sintetizan directamente en el soporte mediante técnicas de fotolitografía, creando una cadena de oligonucleótidos, véase Ilustración 4.

Específicamente, cada *probe sequence* está formada por 25 nucleótidos (25mer), que posteriormente hibridarán con diversos fragmentos de cRNA marcado (*target sequences*).

Ilustración 2. Affymetrix GeneChip

Al tomar un microarray de Affymetrix observamos una matriz de 1.28 cm2 formada por múltiples celdas de 50µm2, como se muestra en la Ilustración 3. Cada una de estas celdas denominadas 'probe cell' contiene alrededor de 10 millones de copias sintetizadas de una misma probe sequence.

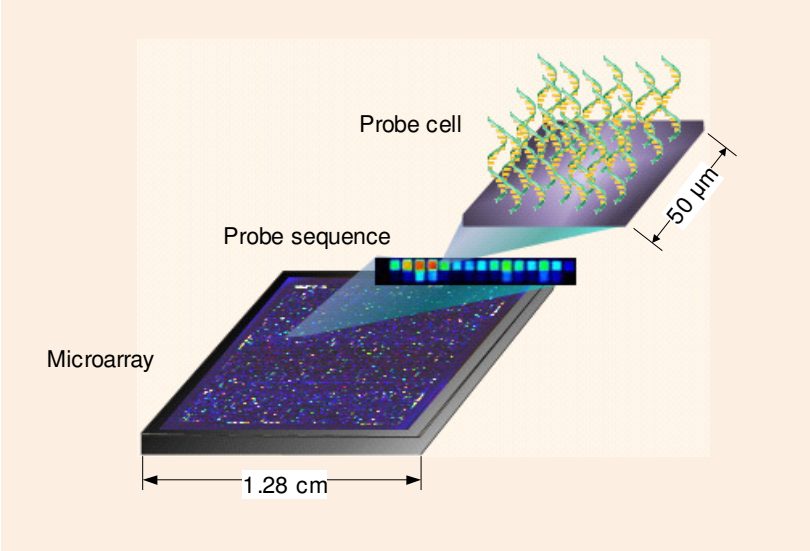


Ilustración 3. Composición de un Microarray

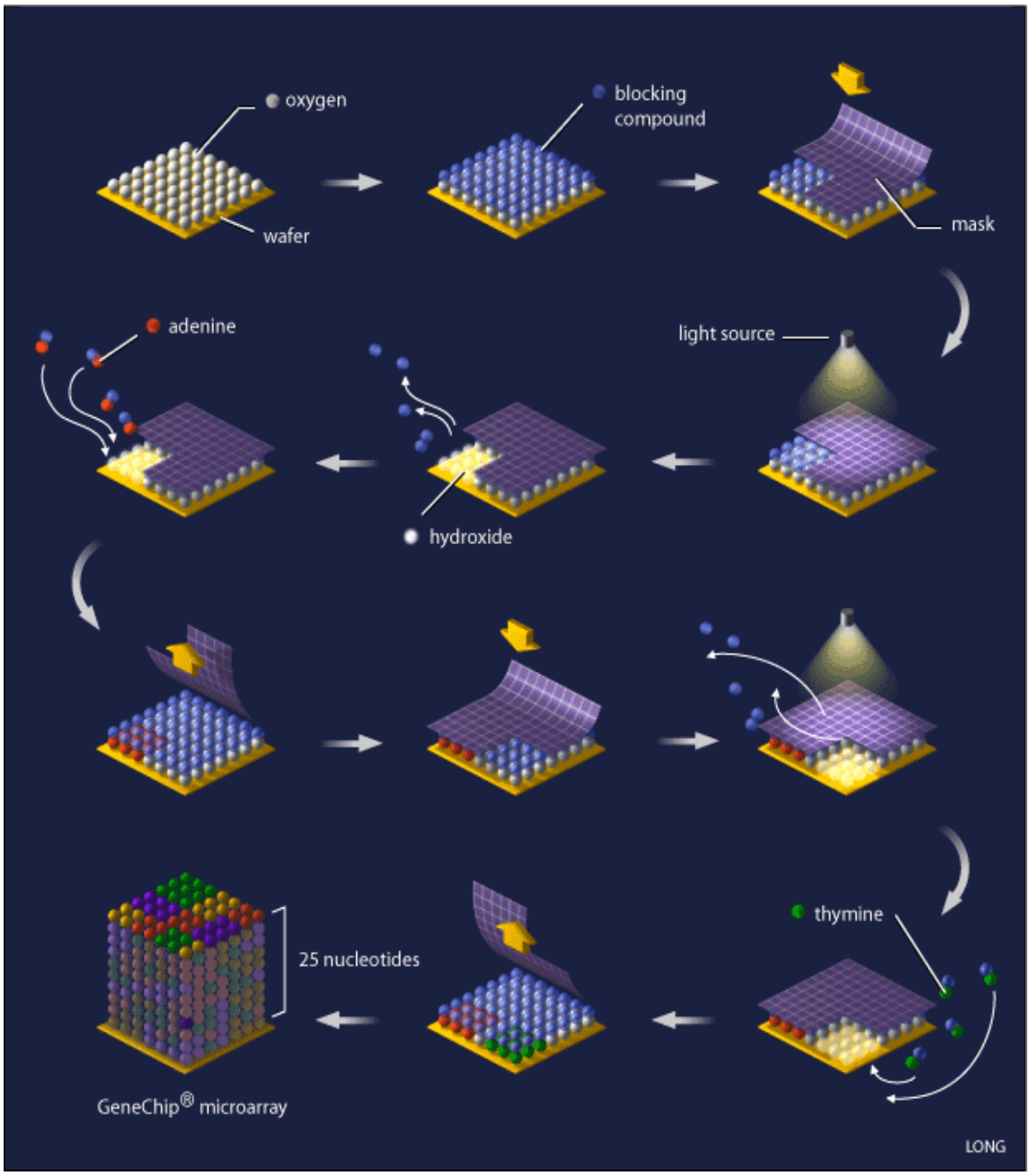


Ilustración 4. GeneChip manufactured

Como sabemos, el fin de un biochip es detectar la presencia y/o ausencia de determinados genes. Para ello se extrae de la célula cierta cantidad de mRNA, se considera a continuación el DNA complementario (cDNA) de la transcripción de dicha secuencia de mRNA.

Para obtener una transcripción, lo más completa posible, lo que se hace es sintetizar un conjunto de unos 9-22 probe cells (11 por defecto) que determinen toda la cadena de mRNA y se denomina probe set. Luego, la cadena de cDNA se etiqueta con biotina y se fragmenta hasta obtener secuencias suficientemente pequeñas para que puedan hibridar con las probe sequences. Posteriormente, después de un proceso de lavado y tinte (washing & stained) se

procede a escanear la imagen resultante.

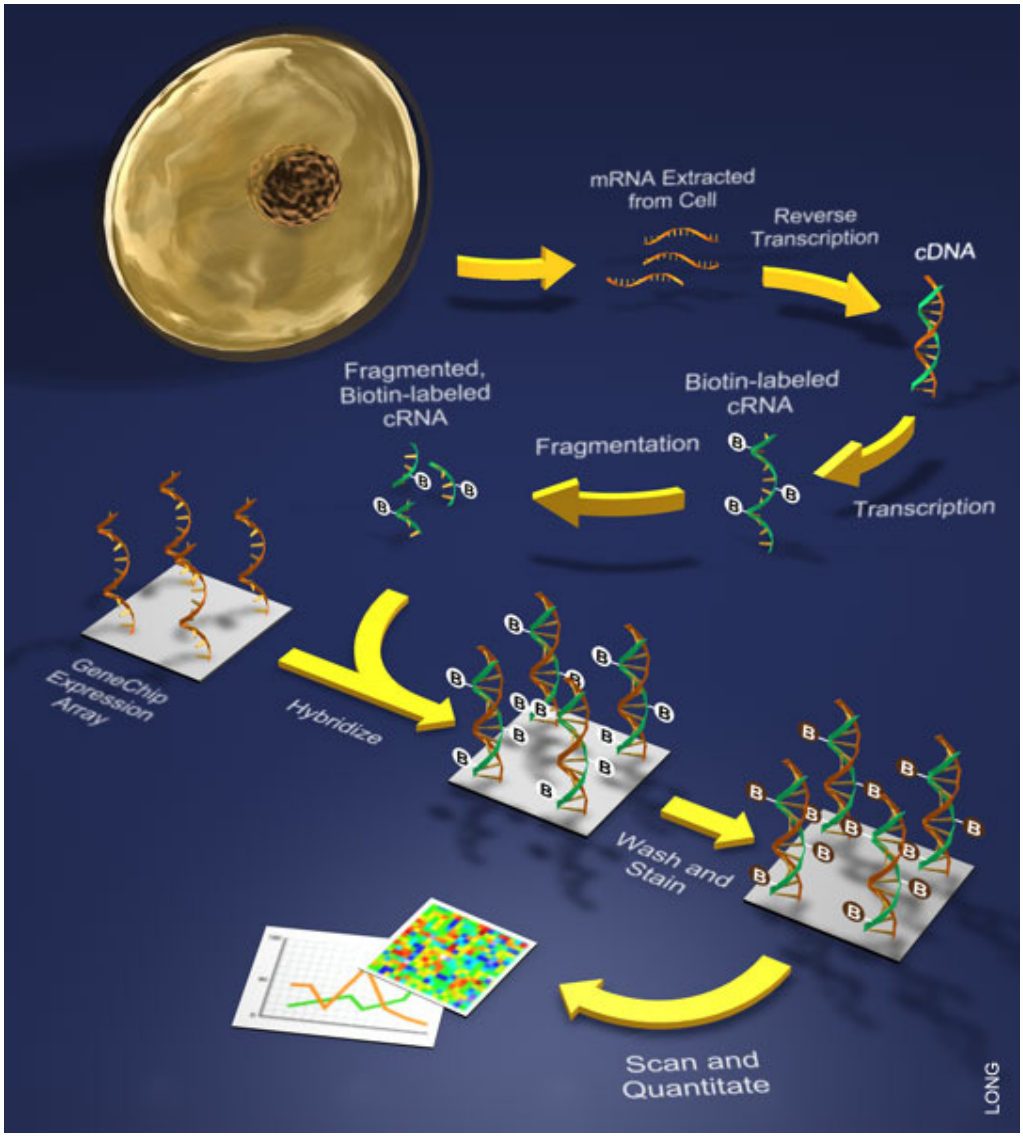


Ilustración 5. Uso de array de oligonucleótidos

Análisis de Arrays

Una vez escaneado el chip obtenemos una imagen digital de las intensidades de cada *probe cell* en estructura matricial que debe ser cuantificada y poder así obtener una medida de la expresión de cada gen.

Especificidad y Sensibilidad de un microarray

Un GeneChip está diseñado para obtener una alta especificidad, esto es, para que sea capaz

de rechazar gran número de 'probe targets' no presentes frente a su complementario en el chip. Sin embargo, no siempre ocurre así, y siempre existe la posibilidad de una hibridación no específica, como por ejemplo, que haya algún gen expresado que no debería.

En la **Tabla** 2 se muestra un cuadro de contraste que marca la sensibilidad y la especificidad de un microarray.

	gen que no debe ser activo	gen que debe ser activo	
gen no activo	VN	FP	Sensibilidad ($\frac{VN}{VN+FP}$)
gen activo	FN	VP	Especificidad ($\frac{VP}{VP+FN}$)

Tabla 2. Especificidad y Sensibilidad

La eficiencia de la hibridación, así como la imagen final resultante dependen de diversos factores, destacando:

- Factores dependientes, como la longitud, extensión de la complementariedad y el desarrollo de la composición base de una secuencia genética.
- Factores independientes, como la relación de concentración entre 'probe sequences' y 'probe targets', el tiempo de exposición, la temperatura, la concentración de cationes, el pH, la media caotrópica y dieléctrica, las características del soporte y la densidad espacial de las moléculas sintetizadas.
- Señal de fondo dependiente de la muestra. Cualquier *probe sequence* tiene el potencial para interactuar con otras secuencias complementarias presentes en la muestra, sobre todo si la complementaria es corta y parcial. Este hecho puede incrementar la señal no específica del *background* y reducir la sensibilidad y la especificidad.
- Hibridación cruzada (cross-hibridization). Este fenómeno se produce cuando ciertas secuencias del cDNA marcado, procedentes de cierta transcripción génica del mRNA, llega a hibridar sobre una secuencia que no le toca.

Todos esos posibles factores se van acumulando a través del proceso de fabricación desencadenando una serie de efectos, o mejor dicho, defectos. Para poder disminuirlos y así aumentar la sensibilidad y la especificidad del chip, Affymetrix introdujo una serie de técnicas destinadas a este fin.

Enfoque Affymetrix

La técnica principal y más característica de un GeneChip es la utilización de un par de secuencias ('*probe pairs*') para cada una de las transcripciones destinadas a ser estudiadas.

La una tiene exactamente la secuencia correspondiente al gen estudiado, llamada Perfect Match Probe (PM) y la otra ha sufrido una inversión en el nucleótido central de la *probe sequence*, tal y como se muestra en Ilustración 6, y recibe el nombre de Mismatch Probe

(MM).

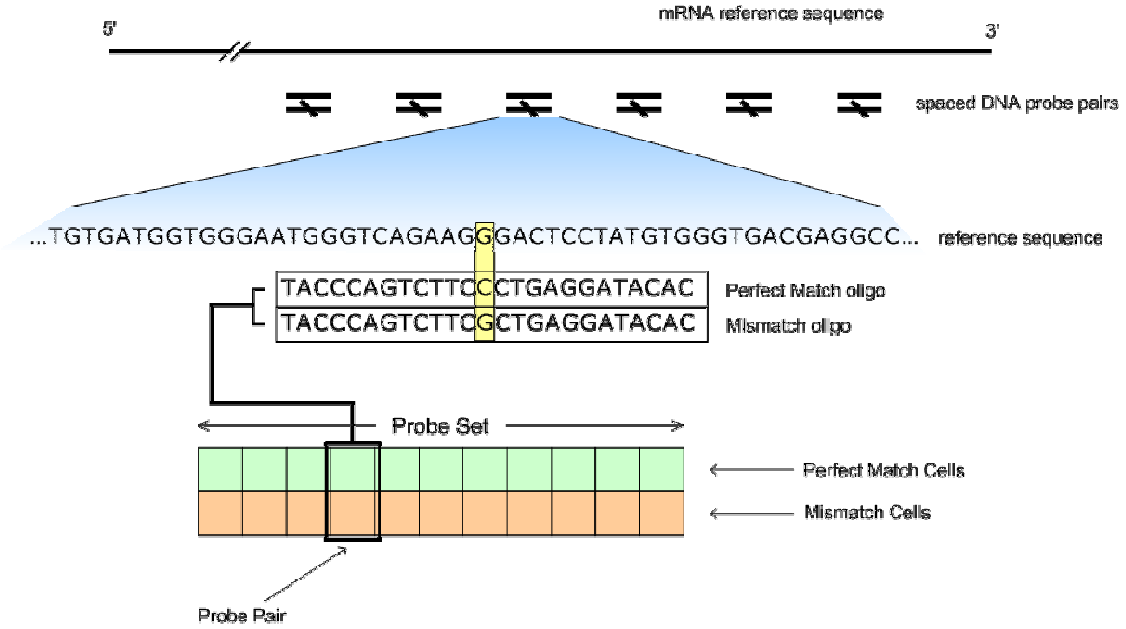


Ilustración 6. Perfect Match and Mismatch Probes Sets

De esta manera se obtiene un balance óptimo entre la sensibilidad y la especificidad del chip.

Las características principales en el uso de de la estrategia PM-MM son:

- PM-MM probe pairs ofrecen mayor sensibilidad cuando las concentraciones de los *probe targets* son bajas.
- Ofrecen también especificidad en la presencia de señales de background complejo.
- Los estadísticos en el análisis de los microarrays, son más precisos con el uso de múltiples *probe pairs*.

Detection Calls

La pregunta que nos hacemos cuando tenemos un biochip delante es: "¿La transcripción de un gen en particular está presente o ausente?"

Para determinar la presencia o ausencia de una transcripción genética para un *probe pair* i se considera el siguiente Factor de Discriminación (*Discrimination Score*), como medida de diferenciación entre las intensidades PM y MM.

$$R_i = \frac{PM_i - MM_i}{PM_i + MM_i}$$

El experimento consiste en calcular la probabilidad de que un R_i observado sea significativamente bajo, i.e., denote ausencia de la transcripción. Para lo cual se establece el siguiente contraste de hipótesis:

$$H_0 : median(R_i) = 0$$
$$H_1 : median(R_i) > 0$$

Sin embargo, para prevenir falsos positivos se ajusta R con un umbral τ (por defecto, $\tau = 0.015$), así el experimento queda en estudiar:

$$H_0 : median(R_i - \tau) = 0$$
$$H_1 : median(R_i - \tau) > 0$$

Notar que un incremento de τ reduciría el nº de falsos positivos detectados, pero también el de verdaderos positivos.

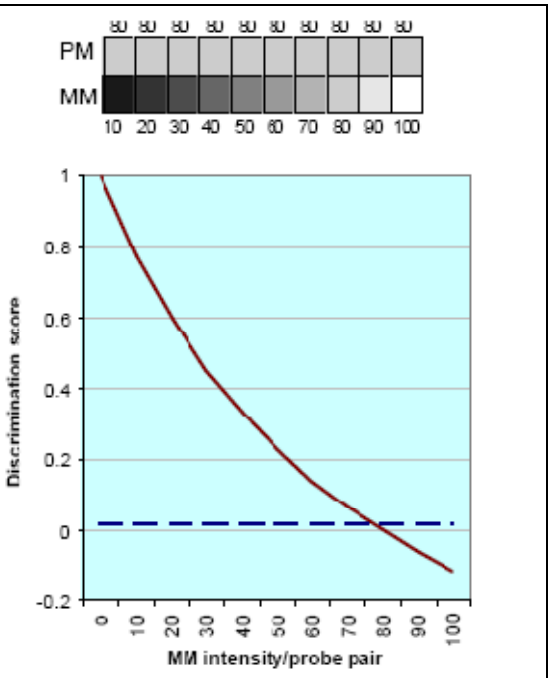


Ilustración 7. Discrimination Score variation

El ejemplo de probe set de la Ilustración 7, muestra la variación de R_i, y hacemos notar que para valores similares de PM y MM, el *Discrimination Score* se mueve en un entorno del cero. La línea discontinua es el valor del umbral τ

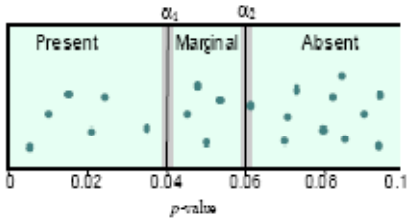
En el estudio, para un valor R_i, se asocia un p-valor concreto (probabilidad de un dato observado bajo condición H_0) que se contrasta mediante *One-Sided Wilcoxon's Signed Rank test*.

Una vez se tienen los p-value de los datos, se consideran dos niveles de significatividad α_1 y α_2 tales que $0 < \alpha_1 < \alpha_2 < 0.5$

$$\alpha_1 = 0.04$$

Por defecto: $\alpha_2 = 0.06$ y según el siguiente criterio, se establece el tipo de observación:

$$\begin{cases} \text{Present Call} & \text{, si } p < \alpha_1 \\ \text{Marginal Call} & \text{, si } \alpha_1 = p < \alpha_2 \\ \text{Absent Call} & \text{, si } p \geq \alpha_2 \end{cases}$$



Significance levels α_1 and α_2 define cut-offs of *p*-values for making calls.

Ilustración 8. Niveles α

Si se reduce α_1 , el nº de falsos *Detected calls* y verdaderos *Detected calls* también se reduce.

Si se incrementa el nivel α_2 , se puede reducir el nº de falsos undetected calls, así como el de verdaderos undetected calls.

Algoritmos de cuantificación de señal (Signal algorithm)

Tratan, sobre una medida cuantitativa de la señal de cada probe set que representa un nivel relativo de expresión de cada transcripción genética.

Motivación:

El aumento exponencial en experimentos de microarrays de ADN de los últimos años ha motivado el desarrollo de muchos Algoritmos de cuantificación de señal (SQ).

Estos algoritmos permiten diversas transformaciones en las medidas reales para permitir a los investigadores a comparar las lecturas de los diferentes genes cuantitativamente dentro de un experimento y a través de experimentos independientes. Sin embargo, no está claro si hay un "mejor" algoritmo para cuantificar los datos de microarrays.

La capacidad de comparar y evaluar este tipo de algoritmos es crucial para cualquier análisis posterior. Se han propuesto diferentes metodologías para comparar los diferentes algoritmos de cuantificación de la señal de datos de expresión génica. Su objetivo ha sido permitir a los investigadores comparar el efecto de diferentes algoritmos SQ en el conjunto de datos específicos.

Se ha combinado dos tipos de pruebas para evaluar el efecto de un algoritmo SQ en términos de relación señal / ruido. Para evaluar el ruido, se ha explorado la redundancia en el conjunto de datos experimentales para comprobar la variabilidad de una salida de un determinado algoritmo SQ. Para el efecto de la SQ en la señal se ha evaluado el exceso de genes expresados diferencialmente utilizando diversas pruebas estadísticas relevantes.

Resultados:

Se ha demostrado el enfoque de análisis de tres algoritmos SQ para microarrays de oligonucleótidos. Se ha comparado los resultados de la utilización del software dChip y el software RMAExpress a los obtenidos utilizando el estándar MAS5 Affymetrix en un conjunto de datos que contienen los pares de hibridaciones repetidas.

El análisis sugiere que dChip es más robusto y estable que las herramientas MAS5 alrededor del 60% de los genes, mientras que RMAExpress es capaz de lograr una mejora aún mayor en

términos de señal a ruido, más del 95% de los genes[1].

En nuestro caso vamos a utilizar MAS5 mencionado anteriormente, es el estándar Affymetrix.

Información Experimental Affymetrix

Dentro de la serie de productos que ofrece Affymetrix para el análisis de microarrays, incluye paquetes de herramientas informáticas que permiten a los investigadores realizar diferentes experimentos basados en sus tipos de microarrays.

Entre esos paquetes, el MicroArray Suite 5.0 (MAS 5.0) fue pionero en el tratamiento y control del proceso de escaneado de los microarrays, así como el análisis y pre proceso de las intensidades de los probe sets. Posteriormente se han desarrollado otros paquetes específicos (véase Tabla 4, Sección

[1] http://www.ncbi.nlm.nih.gov/pubmed/14751998

Software) que se han basado directamente en los algoritmos del MAS 5.0.

La información basada en la imagen del microarray queda registrada en diferentes tipos de archivos dependiendo de su contenido. Para detalles de los tipos específicos véase la Tabla 3.

Data File Type	File extension	Description
Image Data File	*.dat	Image of the scanned probe array and information about the experiment & sample that the image pertains to.
Cell Intensity File	*.cel	Single intensity value for each probe cell delineated by the grid and information about the: 1) image that the cell intensities were derived from, and 2) experiment and sample that the image pertains to.
Probe Analysis Data	*.chp	Includes for each transcript represented on the array, the detection call, change call from comparison analysis and other call metrics.
Experiment Information[2]	*.exp	Information about the experiment name, sample, and probe array type. The experiment name also provides the default name for subsequent data that are generated during experiment analysis.
Library File[2]	*.cdf	Information about the probe array design characteristics, probe utilization and content, and scanning and analysis parameters.
Report File[2]	*.rpt	Text file summarizing data about probe array experiment and quality information for a single experiment. The report is generated from the probe analysis data file.

Tabla 3. Archivos de información de los experimentos

[2] Desde que GCOS pasó a sustituir al MAS, la información de estos archivos ha quedado integrada en los tres primeros de la tabla.

Control de calidad

Los microarrays son fabricados de manera que puedan aportar información del rendimiento de la hibridación y del escaneado de la imagen. Con este propósito, cada uno de ellos contiene un conjunto de *gene probes* que hacen el papel de marcadores de la muestra. El valor de la intensidad que se obtiene de ellos es utilizado para generar los estadísticos del control de calidad del microarray.

Probe array image inspection

Con el objeto de detectar defectos de la imagen, como puedan ser: rascones, contaminación debida al polvo o gotas de humedad o cualquier otro defecto que provoque fallos físicos al *slide* es recomendable realizar una inspección visual de cada uno de los microarrays. En la ilustración podemos observar algunas de estas imperfecciones.

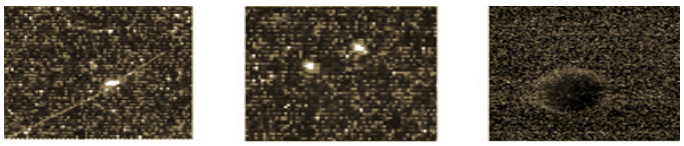


Ilustración 8. Diferentes defectos de imagen

Comparison Analysis (Pre proceso de información)

En un análisis de comparación se toman dos muestras de biochips del mismo tipo y se contrastan las expresiones/intensidades de los genes involucrados en el estudio, para así poder detectar y cuantificar variaciones en dichas expresiones.

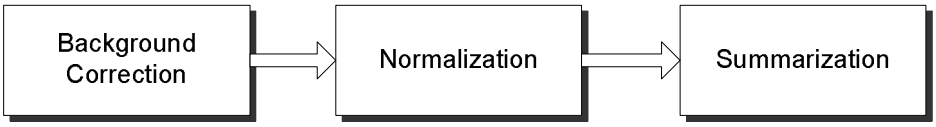
Uno de los arrays se designa como base del estudio (*baseline array*) y el otro como experimento (*experiment array*). En el proceso de análisis se utilizan dos procesos: uno para generar un valor cualitativo del contraste entre arrays, denominado *Change p-value*; el otro nos sirve para hacer una estimación cuantitativa, denominado *Signal Log Ratio*.

En cualquier caso debemos ser conscientes de los experimentos que llevamos a cabo.

Normalmente se procesan múltiples biochips de un mismo estudio con el objeto de obtener las características (genes) que diferencian unos de otros.

El proceso de comparación requiere de un pre proceso de los datos, que permita ajustar los niveles de expresión entre chips.

El pre proceso general de los datos consta de 3 pasos básicos:



Background Correction

¿Qué es la corrección del *background*? Sabemos que existe una cantidad de hibridación no específica contenida en el *background* y que afecta a la sensibilidad y especificidad del chip.

Este paso del pre proceso permite minimizar los efectos que dicha hibridación causa sobre la señal de cada *probe cell.*

Normalización

Previo a la comparación entre dos biochips es necesario aplicar un escalado y/o normalizado para poder ajustar la señal, que puede contener errores causados por factores técnicos y biológicos. ¿Cómo es esto posible? La normalización recoge la información de los probe cell correspondientes a cada probe set de cada chip y los ajusta a un valor comparable, de manera que si alguno esta sobre-expresado o infra-expresado con respecto al total, regulando, así, su valor de expresión génica.

El escalado permite ajustar la intensidad de los *probe sets* tanto del array base, como del array experimento a una intensidad prefijada por el usuario.

Con la normalización, las intensidades de los probe sets del array experimento quedan normalizadas respecto del array base.

Sin embargo, cuando se tratan múltiples arrays se procede a un normalizado común que permita ajustar las intensidades de todos los biochips.

El desarrollo y estudio de las diferentes técnicas de normalización se encuentra en la sección "TÉCNICAS DE NORMALIZACIÓN".

Sumarización

Cada *probe set* que determina una expresión genética está constituido por varios *probe cells*, tal y como sabemos.

Es preciso cuantificar dicha expresión en un valor único (un gen, un valor). La sumarización proporciona para cada *probe set* un valor cuantitativo (*signal*) que representa el nivel relativo de expresión correspondiente a su transcripción genética. Dicha cuantificación se calcula mediante métodos estadísticos que dependen directamente de los *probe cells* que conforman un *probe set* particular.

Dependiendo del tipo de normalización, existen también diferentes formas de sumarizar. Véase, por ejemplo, la sumarización de Affymetrix, que utiliza el 'Signal Algorithm' como método y que, además, lo hace como paso previo a la normalización.

Técnicas de Normalización

Scaling Method

Este método es usado por Affymetrix en su paquete informático de análisis: MAS 5.0 (MicroArray Suite 5.0), aunque actualmente ha pasado a formar parte del GCOS (GeneChip Operating Software)

Background Correction

Notación

En primer lugar el array se divide en K regiones rectangulares (K =16 por defecto), denotadas como Z_k.

Denotamos

$$Z = \left\{ x \in \mathbf{R}^2 \,\middle|\, x \leftarrow \text{array coordenate} \right\}$$

$$Z_k = \left\{ x \in Z \,\middle|\, x \in \text{'region } k\text{'} \right\}$$

Tomamos los valores por debajo del percentil 0.02: $R_k = \left\{ x \in Z_k \,\middle|\, I(x) \le q_{2\%} \right\}$; $k = 1 \ldots K$

Sea $x_k^0 \rightarrow$ centro de la región Z_k.

El valor estimado del background de la región k viene dado por:

$$bZ_k \equiv \overline{I(x)} \qquad ; x \in R_k$$

Y la estimación del ruido presente en el background de la región k es:

$$nZ_k \equiv \sigma^2(I(x)) \qquad ; x \in R_k$$

Formulación

A priori la intensidad ajustada de un probe cell, $A(x)$, se basa en el valor de la intensidad con el valor del background sustraído:

$$A(x) = I(x) - b(x)$$

donde $b(x) \rightarrow$ intensidad estimada de background para el probe cell x

Sin embargo, debemos evitar que la intensidad ajustada se haga negativa, pues los cálculos posteriores del análisis trabajan con transformaciones logarítmicas de los datos. Por tanto, se considera que las intensidades negativas son ruido de fondo, asignando al probe cell en cuestión una expresión basada en nZ_k :

$$A(x) = \max\left\{ I(x) - b(x), \text{NoiseFrac} \cdot \text{n(x)} \right\}$$

donde 'NoiseFrac' es una fracción tomada de la variación global del background, por defecto $NoiseFrac = 0.5$.

Cálculos intermedios

$$b(x) = \frac{\sum w_k(x) \cdot bZ_k}{\sum w_k(x)}$$

$$n(x) = \frac{\sum w_k(x) \cdot nZ_k}{\sum w_k(x)}$$

$$w_k(x) = \frac{1}{\left\| x - x_k^0 \right\|^2 + smooth}$$

(default $smooth = 100$)

Comentarios:

Se asume que el 2% de los datos más pequeños esconde la información total del background.

Sumarización (Signal algorithm)

Signal es calculada como una media robusta de los *probe cells* usando el *One-Step Tukey's Biweight Estimate*.

Formulación

El valor cuantificado para un probe set *j* viene dado por:

$$SignalLogValue_j = Tbw(PV_{j,1}, \ldots, PV_{j,n})$$

donde

$$PV_{j,k} = \log_2(V_{j,k}) \quad , k = 1, \ldots, n$$

$V_{j,k} = \max\{PM_{j,k} - IM_{j,k}, \delta\}$ es el valor ajustado de la señal de cada probe cell *k*, que para garantizar la estabilidad numérica de los métodos se ha considerado $\delta > 0$ suficientemente pequeño, por defecto $\delta = 2^{-20}$

PM-MM Correction

Sabemos que la señal específica del background viene determinada por los MM *probe values*. Si la señal de MM<PM, entonces se puede usar directamente para determinar la expresión del probe pair. En otro caso, es necesario ajustar alguna de las señales para poder ser comparables. En particular, para el Scaling Method, se realiza un ajuste del valor de MM.

$$IM_{k,j} = \begin{cases} MM_{k,j} & ; MM_{k,j} < PM_{k,j} \\ \dfrac{PM_{k,j}}{2^{(SB_k)}} & ; MM_{k,j} \geq PM_{k,j} \wedge SB_k > \mu \\ \dfrac{PM_{k,j}}{2^{\left(\frac{\mu}{1+\left(\frac{\mu - SB_k}{\tau}\right)}\right)}} & ; MM_{k,j} \geq PM_{k,j} \wedge SB_k \leq \mu \end{cases}$$

donde

$SB_k = Tb_k(\log_2(PM_{k,j}) - \log_2(MM_{k,j}))$ → valor medio estimado del background específico mediante un paso del *Tukey bi-weigth test.*

$\mu \rightarrow$ umbral de ajuste del *SB*. Si el SBk es pequeño, se usa mayor señal del PM para determinar el IM. ($\mu = 0.03$ by default)

$\tau \rightarrow$ este umbral describe la variabilidad de los probe pairs en el probe set correspondiente. ($\tau = 10$ by default)

Normalización

El ajuste de normalización de los valores de la señal tanto de cada probe pair, como de los probe set, viene dado por dos factores multiplicativos, uno de escalado y otro de normalizado.

El escalado ajusta cada array usando una media truncada al 2% de entre todas las señales de los probesets:

$$sf = \frac{Sc}{2^{\overline{SignalLogValue_j}_{2\%}}}$$

^{Sc} → User target intensity (500 by default)

El factor de normalización utiliza un array base para ajustar la señal de todos los array sobre aquel:

$$nf = \frac{\overline{SPVb_{j,k}}_{2\%}}{\overline{SPVe_{j,k}}_{2\%}}$$

$SPV_{j,k} \rightarrow$ SignalProbeValue probe pair k in probe set j

$b \rightarrow$ indica baseline probe set

$e \rightarrow$ indica experimente probe set

En definitiva, para un probe set *j*, el valor normalizado viene dado por:

$$RV_j = nf \cdot sf \cdot 2^{SignalLogValue_j}$$

y para un probe pair k

$$SPV_{j,k} = PV_{j,k} + \log_2(nf * sf)$$

Observaciones:

Esta fórmula se encuentra en escala logarítmica de base 2, la alternativa real es:

$$SV_{j,k} = nf \cdot sf \cdot V_{j,k}$$

Otros ajustes:

Los factores de normalización y escalado pueden venir definidos por el usuario. Y en el caso de buscar un ajuste absoluto, el factor de normalización se establece en 1, que viene a ser, simplemente, un escalado.

Robust Multiarray Average (RMA)

Generalmente los métodos de normalización utilizan uno de los arrays como base de la normalización, en RMA no ocurre tal cosa, sino que todos los chips son tratados como iguales, no se usa un chip como referencia en la normalización.

El método de normalización asume que las intensidades de los chips siguen una distribución común. Podemos comparar las distribuciones de las intensidades de los chips de dos en dos visualizando un `qqplot`[3]. Así, ambas distribuciones son más similares cuanto más se ajusta la gráfica del `qqplot` a la diagonal.

La idea matemática de un ajuste a la diagonal entre dos distribuciones, consiste en proyectar el valor de normalización sobre la recta $y = x$, dada por el vector $\left(1/\sqrt{2}, 1/\sqrt{2}\right)$. El método de RMA es una generalización de dicha idea a dimensión N.

Consideremos la siguiente nomenclatura para un conjunto de microarrays:

$i \rightarrow$ arraychip

$n \rightarrow$ probe set

$j \rightarrow$ probe pair in probe set

Background Correction

Se considera que los datos de las intensidades de los *perfect match* siguen un modelo de sumación entre la señal del background, Bg, y la señal real, s:

$$PM_{ijn} = Bg_{ijn} + s_{ijn}$$

También asumimos que el nivel de background medio es común a cada array, es decir: $E(s_{ijn}) = \beta_i$

Una primera aproximación al valor ajustado de PM viene dado por:

$$PM_{ijn} - \beta_i$$

donde $\log_2(\beta_i)$ = mode of $\log_2(MM)$ distribution

En la práctica, si la concentración de pm es relativamente pequeña, ocurre que $PM_{ijn} < \beta_i$, lo cual puede ser un problema cuando tomamos una transformación logarítmica de la ecuación $PM_{ijn} - \beta_i$. Por esta razón, la alternativa que se considera para la corrección del background es tomar $B(PM_{ijn}) = E(s_{ijn} \mid PM_{ijn})$

[3] el QQ Plot (o relación de quantiles) para dos conjuntos de datos dados, muestra la comparación entre la distribución de un conjunto enfrentada a la distribución del otro conjunto.

Normalización

Como hemos asumido que las intensidades de la señal de cada uno de los chips siguen una misma distribución, el algoritmo que viene a continuación es el que seguiremos para conseguir dicha hipótesis:

1) Sea X la matriz pxN, donde p es el número valores de las intensidades de cada uno de los PM en cada chip y N el número de chips.

2) Sea X_{sort} la matriz X donde los elementos de cada columna han sido ordenados de menor a mayor.

3) Si $q_i = (q_{i1}, \ldots, q_{iN})$ la fila *i* de la matriz X_{sort}, entonces el valor de normalización viene dado por

$$\frac{\langle q_i, d \rangle}{\|d\|^2}$$

donde $d = \left(1/\sqrt{N}, \ldots, {}_{N}, 1/\sqrt{N} \right)$, correspondiente al vector director de la recta $x_1 = x_2 = \ldots = x_N$

En definitiva la fila *i* de la matriz X vendrá dada por

$$\text{proj}_d q_i = \frac{\langle q_i, d \rangle}{\|d\|^2} d = \frac{1}{\sqrt{N}} \sum_j q_{ij} d = \left(\frac{1}{N} \sum_j q_{ij}, \ldots\ldots, {}_{N}, \frac{1}{N} \sum_j q_{ij} \right)$$

4) La matriz obtenida después de este proceso, denotada por X'_{sort}, se le devuelve su ordenación original, para obtener así la matriz normalizada final: X_{norm}

Sumarización

Ahora tenemos una matriz de datos X con el ajuste del background y la normalización realizada. De dicha matriz podemos considerar los datos Y_{ijn}, que vienen siendo, para cada probe set *n*, los valores de las intensidades de los PM con el background ajustado, normalizados y en base logarítmica.

$$Y_{ijn} = \log_2 (Norm(PM_{ijn} - BG_i))$$

Por otro lado, si S_{ijn} es la intensidad del probe pair *j*, del chip *i* para el probe set *n*. Podemos asumir que dicha intensidad es un múltiplo de la cantidad del gen expresado por el probe set *n* en el chip *i* y la afinidad de los probepairs *j* correspondientes a todos los chips para cada probe set *n*.

$$S_{ijn} \propto f_{in} a_{jn}$$

donde

$f_{in} \rightarrow$ cantidad de gen n expresado en el chip i

$a_{jn} \rightarrow$ afinidad de los probepairs j del probe set n

Así obtenemos que cada Y_{ijn} sigue el siguiente modelo:

$$Y_{ijn} = \mu_{in} + \alpha_{jn} + \varepsilon_{ijn}$$

donde

$\mu_{in} = \log(f_{in})$, $\alpha_{jn} = \log(a_{jn})$ y ε_{ijn} representa una variable de error independiente e idénticamente distribuida con media 0.

Por tanto el conjunto $\{\mu_{in}\}_{i=1}^{N}$ son los valores de expresión del probeset n para cada uno de los chips. A partir de los cuales podemos obtener el valor de expresión, μ_n, del gen representado por el probeset n, mediante una estimación robusta, normalmente un *'median polish'*.

Software

Toolkit Name	Description
MAS	Affymetrix MicroArray Suite provides instrument control for the GeneChip® Scanner 3000 or GeneArray® 2500 Scanner and the GeneChip Fluidics Station 400. MAS also provides array image acquisition and analysis for all GeneChip® arrays, and provides the end user interface for the Affymetrix Lims software for data storage and management. All MAS functionality can be quickly and easily accesses using multiple toolbar displays, enabling you to rapidly locate and analyze your GeneChip array data. Note: The Microarray Suite Software has been discontinued. It is no longer supported by Affymetrix. This product has been replaced by GeneChip® Operating Software (GCOS).
GCOS	Affymetrix GeneChip® Operating Software (GCOS) automates the control of GeneChip® Fluidics Stations and Scanners. In addition, GCOS acquires data, manages sample and experimental information, and performs gene expression data analysis. GCOS also supports the GeneChip® DNA Analysis Software (GDAS), GeneChip® Genotyping Analysis Software (GTYPE), and GeneChip® Sequence Analysis Software (GSEQ) for resequencing and genotyping data analysis.
AGCC	Affymetrix GeneChip Command Console™ Software (AGCC) will be the replacement for the GeneChip Operating Software (GCOS, GCOS Server). AGCC will provide features for sample and array registration, data management, fluidics and scanning instrument control, automatic and manual image gridding, and summarizes Probe Cell intensity data (*.CEL file generation). Probe Level Summarization (*.CHP files) will be generated by other Affymetrix® software and tools.
Affymetrix® Expression Console™	Affymetrix® Expression Console™ software provides an easy way to create summarized expression values (CHP files) for individual or collections of 3' Expression Array and Exon Array feature intensity (CEL) files. In addition to CHP writing, the Expression Console application also produces a collection of QC metrics for evaluating the success of hybridizations.
Bioconductor Project	Bioconductor is an open source and open development software project for the analysis and comprehension of genomic data. Bioconductor is primarily based on the R programming language but we do accept contributions in any programming language. There are two releases of Bioconductor every year (they appear shortly after the corresponding R release). At any one time there is a release version, which corresponds to the released version of R, and a development version, which corresponds to the development version of R. Most users will find the release version appropriate for their needs. In addition there are a large number of meta-data packages available. They are mainly, but not solely oriented towards different types of microarrays.
DNA Chip Analyzer (dChip)	DNA-Chip Analyzer is a Windows software package for probe-level and high-level analysis of Affymetrix gene expression microarrays and SNP microarrays. Gene expression or SNP

	data from other microarray platforms can also be analyzed by importing as external dataset. At the probe level, dChip can display and normalize the CEL files, and the model-based approach allows pooling information across multiple arrays and automatic probe selection to handle cross-hybridization and image contamination. High-level analysis in dChip includes comparing samples, hierarchical clustering, view expression and SNP data along chromosome, LOH and copy number analysis of SNP arrays, and linkage analysis. In these functions the gene information and sample information are correlated with the analysis results.
GenePattern	GenePattern provides access to many of the computational methods used to analyze genomic data. Using GenePattern, you can easily run analyses, visualize results, and build pipelines to recreate analytic processes that use a combination of tools. Its extendable architecture makes it easy for computational biologists to add analysis and visualization modules, which ensures that you have access to new computational methods on a regular basis. Note: GenePattern has replaced the GeneCluster software.
GeneTraffic	**GeneTraffic** is a client/server system for analysis and visualization of Affymetrix or Two Color arrays as well as a central database for data storage.

Tabla 4. Software disponible para el análisis de Microarray

Bioconductor project

Bioconductor[4] nace como un proyecto de desarrollo programación libre destinado al análisis y comprensión de datos genéticos.

Principalmente está basado en el lenguaje de programación R[5] preparado especialmente para el análisis estadístico.

La potencia de R y bioconductor son los paquetes, soportes de funciones, datos y documentos que pueden ser descargados e instalados a través de la red.

[4] http://www.bioconductor.org/

[5] http://www.r-project.org/

Bioconductor Packages

En la fase de pre proceso de los genechips de affymetrix seran necesarios los siguientes paquetes[6]:

Paquetes	Descripción
reposTools	Repository tools for R
Biobase	Biobase: Base functions for Bioconductor
annotate / AnnBuilder	Bioconductor annotation data package builder
Affy	Methods for Affymetrix Oligonucleotide Arrays
Simpleaffy	Very simple high level analysis of Affymetrix data

Tabla 5. Paquetes affymetrix

Diagnostic Plots & Quality Analysis

Image plots

Poder observar la imagen de los chips nos permite detectar posibles fallos en el escaneo, como ruido del bias, bloques dañados, etc.

La función:

```
image(rawData[,i])
```

nos muestra la imagen compuesta para el chip i, en blanco y negro, si queremos algo más colorido podemos utilizar:

```
image (rawData[,i], col=rainbow(512))
```

Podemos guardar la imagen resultante con la función:

```
> dev2bitmap("rawData.Image.jpg",type="jpeg")
image(rawData[,i])
```

nos muestra la imagen compuesta para el chip i, en blanco y negro, si queremos algo más colorido podemos utilizar:

```
image (rawData[,i], col=rainbow(512))
```

Podemos guardar la imagen resultante con la función:

```
> dev2bitmap("rawData.Image.jpg",type="jpeg")
```

[6] El listado de paquetes se puede encontrar en la dirección: http://www.bioconductor.org/repository/

Histograms & BoxPlot

La representación de los histogramas y los boxplot comparan las distribuciones que siguen cada uno de los chips.

```
> hist(rawData)
> boxplot(rawData, col=n)
```

siendo `n` el número de muestras.

Nota: cuando la función hist actúa sobre un objeto abatch nos representa la distribución de las intensidades de los PM en escala logarítmica.

RNA Degradation Analysis

```
> RNAdeg.Data <- AffyRNAdeg(rawData)
> plotAffyRNAdeg(RNAdeg.Data)
```

Quality Control

```
> library(simpleaffy)
> qc.data <- qc(rawData)
> avbg(qc.data)
```

Proporciona los valores medios del background de todos los arrays, dichos valores deben ser magnitudes comparables.

```
> sfs(qc.data)
```

Scale Factors de los arrays.

```
> ratios(qc.data)
```

Relación media entre los extremos de cada probe sequence(5'/3'). Dichos valores no deben ser superiores a 3.

Low-Level Analysis

Cada uno de los métodos estudiados tiene técnicas de pre procesamiento diferentes y de ese mismo modo lo entiende el software de R, aunque las funciones principales son las mimas para todos ellos.

Background Correction

```
> bgData <- bg.correct(rawData,"method")
```

donde los métodos figuran en la tabla 6:

Método	Opciones
MAS 5.0	`mas`
RMA	`rma`
GCRMA	`rma2`
Loess	`none`
LiWong	`none`
VSN	`none`

Tabla 6. Métodos de background correction

Normalización

```
> normData <- normalize(bgData,"method")
```

Las opciones posibles para cada método se recogen en la tabla:

Método	Opciones
MAS 5.0	constant
RMA	quantiles
GCRMA	
Loess	loess
LiWong	invariantset
VSN	vsn

Tabla 7. Opciones de los métodos de background correction

Summarizing

```
> esetData <- computeExprSet(normData, pmcorrect.method,
summary.method)
```

Las opciones posibles de la variable de sumarización se recogen en la tabla siguiente:

Técnica	Opciones
MAS 5.0	`mas`
RMA	`medianpolish`
GCRMA	
Loess	`playerout`
LiWong	`liwong`
VSN	`medianpolish`

Tabla 8. Opciones de sumarización

Para las posibilidades de la corrección del PM, tenemos:

Métodos	Opciones
MAS 5.0	`mas`
RMA[7]	`pmonly`
GCRMA	`gcrma`
Loess	`subtractmm (?)`
LiWong	`none`
VSN	`none`

Tabla 9. Opciones de la corrección PM

La variable 'eSetData' es un objeto de tipo exprSet y se encarga de contener la expresión de cada gen en cada una de la muestras. Es el último eslabón del pre proceso.

Finalmente podemos observar los efectos de la normalización volviendo a plotear los gráficos MA, pero esta vez el tiempo de computación es menor, puesto que la sumarización ha reducido el número de puntos que se contrastan.

[7] La técnica de RMA no realiza ningún tipo de corrección PM, sin embargo debemos indicar al proceso que dicha técnica sólo utiliza la información de los PM.

Materiales

Base de datos

La base de datos contiene casos de tumores cerebrales biopsiados y con Microarrays Affymetrix de expresión genética.

El tipo de microarray es: HG-U133_Plus_2.

La cantidad total de genes que contiene la base de datos es de 54675 genes.

Las muestras están diagnosticadas según la taxonomía de Tumores del Sistema Nervioso Central creada por la World Health Organization (WHO).

Una vez pre procesados los datos, obtenemos el objeto exprdata de tipo ExpresionSet. Mediante la función varLabels podemos observar como están organizados los datos. A continuación mostraremos los más relevantes para la realización de nuestro trabajo:

```
> varLabels(exprdata)
  [1] "CASE.CODE"
```

Código del microarray.

```
  [2] "CENTRE.ID"
```

Centro donde se ha hibridado el microarray.

```
  [3] "CASE.FILENAME"
```

Nombre del fichero donde se encuentra el microarray.

```
 [12] "CR.CONSENSUS.CLINICAL.DIAGNOSIS"
```

Nombre del tumor consensuado por un grupo de expertos.

Existen diversas opciones a la hora de organizar toda esta información contenida en la base de datos. Necesitábamos para los experimentos, organizar en grupos de diferentes tumores diagnosticados. Para ello utilizamos la siguiente convinación de comandos en R:

```
length(which(Nombre_de_grupo_tumoral)== phenoData(exprdata)$LowGlialLabels))
```

En el lugar de la variable: `Nombre_de_grupo_tumoral`, situaremos el nombre del grupo del cual queremos saber el número total de casos que se dan en la base de datos.

Experimentos	Número de casos
Gliales bajos	
`length(which(AS2 == phenoData(exprdata)$LowGlialLabels))`	20
`length(which(OLIGO == phenoData(exprdata)$LowGlialLabels))`	12
Gliales Anaplásticos (GG3) frente a Glioblastomas (GBM)	
`length(which(GBM == phenoData(exprdata)$AnaGlialLabels))`	38
`length(which(GG3 == phenoData(exprdata)$AnaGlialLabels))`	10
Glioblastomas (GBM) frente al resto de tipos tumorales	
`length(which(GBM == phenoData(exprdata)$GlioLabels))`	38
`length(which(OTHER== phenoData(exprdata)$GlioLabels))`	54
Gliales altos (HGG) frente a Gliales bajos (GG2)	
`length(which(GG2 == phenoData(exprdata)$HighLowGlialLabels))`	32
`length(which(HGG == phenoData(exprdata)$HighLowGlialLabels))`	48
Clasificación de grados: GG2= (AS2,OA2,OD2) ,GG3= (AS3,OA3,OD3) y GBM	
`length(which(GG2 == phenoData(exprdata)$GradosLabels))`	32
`length(which(GBM == phenoData(exprdata)$GradosLabels))`	38
`length(which(GG3 == phenoData(exprdata)$GradosLabels))`	10

Tabla 10. Número de casos por experimento realizado

En los experimentos cuyo número total de casos sea mayor, esperamos unos resultados con mayor calidad.

Experimentos

Conjunto de datos

- Casos de tumores cerebrales biopsiados y con Microarrays Affymetrix de
- expresión genética.
- Tipo de microarray: HG-U133_Plus_2.
- Las muestras están diagnosticadas según la taxonomía de Tumores del
- Sistema Nervioso Central creada por la World Health Organization (WHO).
- En la carpeta asociada están los ficheros que contienen los microarrays con los que vamos a trabajar.

Preparación de las etiquetas

- La variable Labels del atributo phenoData contiene los diagnósticos histopatológicos de las muestras: phenoData(exprdataGlial234)$Labels.
- Hemos Modificado las etiquetas de clases mapeando los diagnósticos histopatológicos a etiquetas de clases que agrupan varios diagnósticos. Por ejemplo la etiqueta AS2Set contiene los DIFFUSEASTROCYTOMA9400/3 y FIBRILLARYASTROCYTOMA9420/3.

Algunas veces no existe una etiqueta de muestra en LowGlialList, en esos casos la sustituiremos por "other" con la función Fusiona.

Clasificación, evaluación y selección

Clasificación y evaluación de tumores cerebrales por firmas genéticas

Hay diferentes tipos de tumores cerebrales. Hemos hecho diferentes agrupaciones y aplicado a diferentes componentes principales para poder comparar resultados. También se ha analizado cual es la tendencia del número de genes supervisados a utilizar para obtener la menor tasa de error de mala clasificación.

Selección del clasificador óptimo

Para la clasificación hemos utilizado dos métodos diferentes en la selección de variables: KRSTT y Hold-one-out En función de los resultados podremos decir cual de los dos es más apropiado para nuestro conjunto de datos.

Elavoración de un Top10 genético

Por último, hemos estudiado cuales son los genes que mayor número de veces son seleccionados por el clasificador, tanto para KRSTT como para Hold-one-out.

A continuación vamos a explicar qué pasos hemos seguido para obtener todos estos resultados.

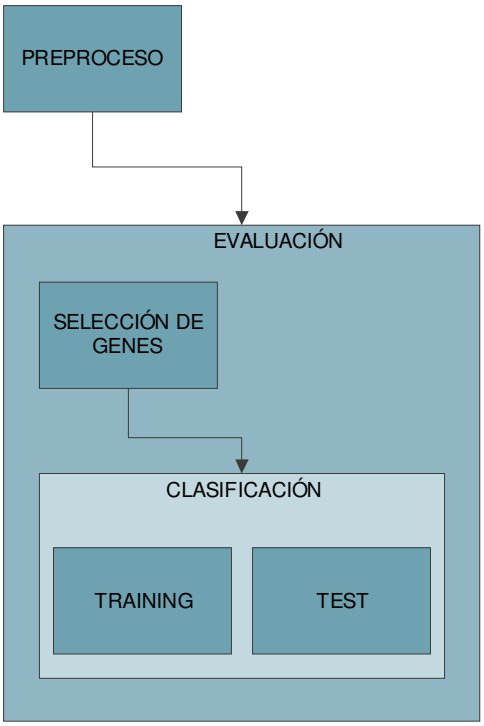


Diagrama 1. Metodología de experimentación de la expresión genética, con bucles anidados de entrenamiento y selecciones de genes/modelos para evitar subestimaciones del error.

Para el caso de los Gliales Bajos AS2 vs. OLIGO (OA2 vs.OD2) la variable CR.CONSENSUS.CLINICAL.DIAGNOSIS del atributo phenoData contiene el diagnóstico del tipo de tumor consensuado por varios médicos. Vamos a clasificar sólo los Glioblastomas, en principio se intentó comparar entre los tres grupos principales: AS2, OA2 y OD2. Pero debido a los pocos casos de OA2 y OD2 los hemos unidos bajo una misma etiqueta, por lo tanto estarán etiquetados: como AS2 los DIFFUSEASTROCYTOMA9400/3 y FIBRILLARYASTROCYTOMA9420/3, y como OLIGO los OLIGOASTROCYTOMA9382/3 y OLIGODENDROGLIOMA9450/3.

Hacemos una búsqueda de estos tipos de tumor en toda la base de datos, y los pre procesamos. La librería affy incluye las funciones Expresso, RMA (Robust Multiarray Average) y justRMA, entre otras, que facilitan el pre proceso completo de un conjunto de microarrays. Estos métodos convierten el objeto de tipo AffyBatch en un objeto de tipo ExpressionSet. Hemos elegido el método JustRMA por ser el más eficaz.

Una vez ya tenemos el objeto del tipo ExpressionSet, vamos a seleccionar las muestras para

el Training y para el Test, que introduciremos en nuestro clasificador.

Para la selección de genes diferenciados por expresión hemos utilizado dos métodos diferentes como ya se ha mencionado antes. Los métodos son el Boostrap, método de estimación basado en el procedimiento estadístico de muestreo con reemplazo y el Hold-one-out, método basado en la validación cruzada. En este caso separaremos aleatoriamente un 70% de los casos para el training del clasificador y el 30% restante para el test. Todo esto se realizará en un bucle de unas 200 repeticiones, para que los resultados tengan un mínimo de calidad, aplicando una metodología k-random sampling training-test (kRSTT).

Al final del bucle tenemos varias variables en las que acumulamos estadísticas relacionadas con el acierto del clasificador. Tenemos un medidor para cada una de las clases de tumor elegidas para el experimento. La evolución mediante Acc nos indicará como de bueno es el clasificador y otra con el acierto marginal o BAR (Balance accuracy rate), es decir, como de sensible/robusto es nuestro clasificador.

Más experimentos que hemos realizado variando las componentes principales supervisadas:

Experimentos realizados
Gliales Bajos AS2 frente a Gliales bajos OA2 y OD2
Gliales Anaplásticos GG3 = (AS3, OA3, OD3) frente a Glioblastomas (GBM)
Glioblastomas (GBM) frente al resto de tumores
Clasificación de Gliales bajos GG2= (AS2, OA2, OD2) frente a Gliales altos HGG = (GBM, GG3= (AS3, OA3, OD3))
Clasificación de grados: GBM vs. GG2 = (AS2, OA2, OD2) vs. GG3 = (AS3, OA3, OD3)

Tabla 11. Experimentos realizdos

Resultados

A continuación, podemos ver los resultados de clasificar y evaluar tumores cerebrales por firmas genéticas. Han sido evaluados de acuerdo con las siguientes medidas de rendimiento:

Acierto: Es la probabilidad de clasificar correctamente un gen para una clase dada.

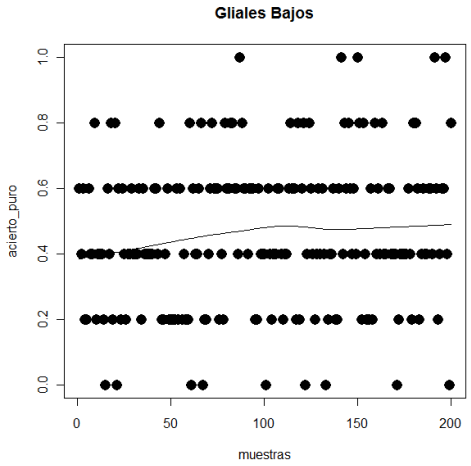
Acierto_marginal (BAR): Es la media de las probabilidades de acierto en cada clase.

Acierto_claseX: Es la probabilidad que tiene un gen seleccionado al azar de pertenecer a esa clase X.

Clasificación de Gliales bajos con 10 genes. Método kRSTT.

Los Gliales Bajos tipo AS2 han sido codificados como CLASE1.

El resto de Gliales Bajos han sido codificados como CLASE2.



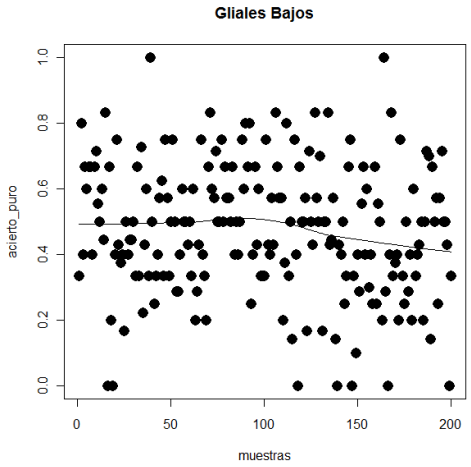
	1st Qu	Mean	3rd Qu
acierto_clase1	0.3333	0.5833	0.6667
acierto_clase2	0.00	0.28	0.50
acierto	0.400	0.462	0.600
BAR	0.3333	0.4317	0.5833

Tabla 12. Tasas de acierto para la clasificación de Gliales Bajos kRSTT

Genes	212284_x_at	212869_x_at	211943_x_at	207783_x_at	AFFX-hum_alu_at
Veces seleccionados	130	128	104	85	83
Genes	1553588_at	211445_x_at	213477_x_at	214003_x_at	200714_x_at
Veces seleccionados	78	66	57	47	44

Tabla 13. Genes seleccionados un número mayor de veces con kRSTT

Método Hold-one-out



	1st Qu	Mean	3rd Qu
acierto_clase1	0.500	0.599	0.750
acierto_clase2	0.0000	0.2633	0.50
acierto	0.3333	0.4742	0.6000
acierto_marginal	0.2812	0.4311	0.5417

Tabla 14. Gliales Bajos Hold-one-out

Genes	AFFX-hum_alu_at	212869_x_at	212284_x_at	211445_x_at	211943_x_at
Veces seleccionados	108	101	99	99	95
Genes	207783_x_at	218601_at	200714_x_at	1553588_at	214585_s_at
Veces seleccionados	78	74	69	66	55

Tabla 15. Genes seleccionados un número mayor de veces con Hold-one-out

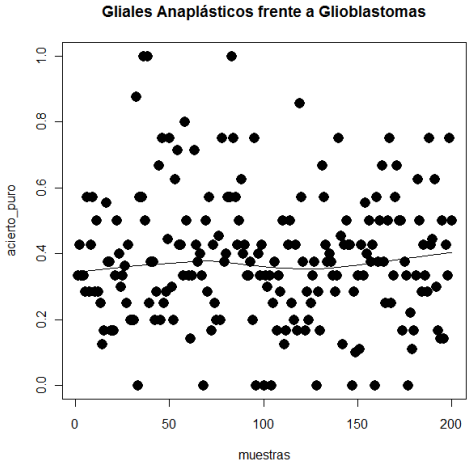
Clasificación de Gliales Anaplásticos frente a Glioblastomas con 10 genes. Método KRSTT.

Numero total de casos de Glioblastomas = 38

Los Glioblastomas han sido codificados como CLASE1

Numero total de casos de Gliales Anaplásticos = 10

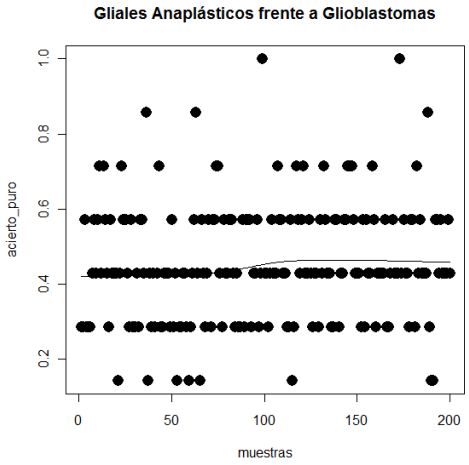
Los Gliales Anaplásticos han sido codificados como CLASE2



	1st Qu	Mean	3rd Qu
acierto_clase1	0.2500	0.4713	0.6667
acierto_clase2	0.200	0.2863	0.400
acierto	0.2500	0.3844	0.5000
acierto_marginal	0.2500	0.3789	0.5000

Tabla 16. Gliales Anaplásticos frente a Glioblastomas KRSTT

Método Hold-one-out



	1st Qu	Mean	3rd Qu
acierto_clase1	0.5000	0.4713	0.6667
acierto_clase2	0.200	0.323	0.400
acierto	0.2857	0.4543	0.5714
acierto_marginal	0.4500	0.5527	0.7000

Tabla 17. Gliales Anaplásticos frente a Glioblastomas Hold-one-out

Genes	213477_x_at	201429_s_at	1553588_at	211296_x_at	201492_s_at
Veces seleccionados	120	112	104	101	97
Genes	213583_x_at	204892_x_at	212284_x_at	208695_s_at	208825_x_at
Veces seleccionados	91	75	69	66	54

Tabla 18. Genes seleccionados un número mayor de veces con Hold-one-out

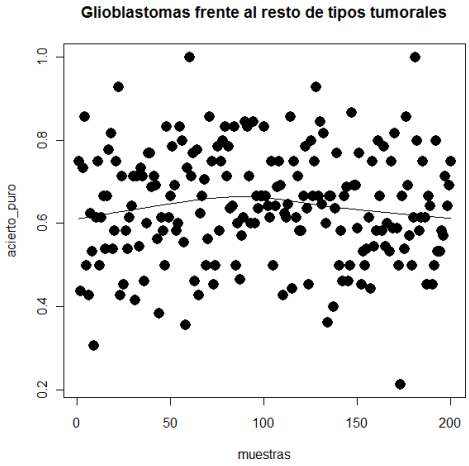
Clasificación de Glioblastomas frente al resto de tipos tumorales con 10 genes. Método KRSTT.

Numero total de casos de Glioblastomas = 38

Los Glioblastomas han sido codificados como CLASE1

Numero total de casos del resto de tipos tumorales = 54

El resto de tipos tumorales han sido codificados como CLASE2



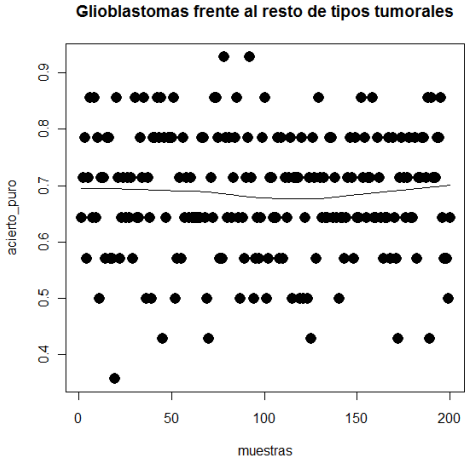
	1st Qu	Mean	3rd Qu
acierto_clase1	0.5556	0.6435	0.7778
acierto_clase2	0.5000	0.6462	0.8250
acierto	0.5655	0.6413	0.7500
acierto_marginal	0.5556	0.6447	0.7639

Tabla 19. Glioblastomas frente al resto de tipos tumorales KRSTT

Genes	213477_x_at	212284_x_at	207783_x_at	1553588_at	212869_x_at
Veces seleccionados	133	127	120	107	105
Genes	214327_x_at	201492_s_at	208834_x_at	AFFX-hum_alu_at	211943_x_at
Veces seleccionados	92	80	74	72	71

Tabla 20. Genes seleccionados un número mayor de veces con KRSTT

Método Hold-one-out



	1st Qu	Mean	3rd Qu
acierto_clase1	0.6389	0.6961	0.7778
acierto_clase2	0.600	0.662	0.800
acierto	0.6429	0.6839	0.7857
acierto_marginal	0.5889	0.6791	0.7778

Tabla 21. Glioblastomas frente al resto de tipos tumorales Hold-one-out

Genes	213477_x_at	212284_x_at	207783_x_at	1553588_at	212869_x_at
Veces seleccionados	133	127	120	107	105
Genes	214327_x_at	201492_s_at	208834_x_at	AFFX-hum_alu_at	211943_x_at
Veces seleccionados	92	80	74	72	71

Tabla 22. Genes seleccionados un número mayor de veces con Hold-one-out

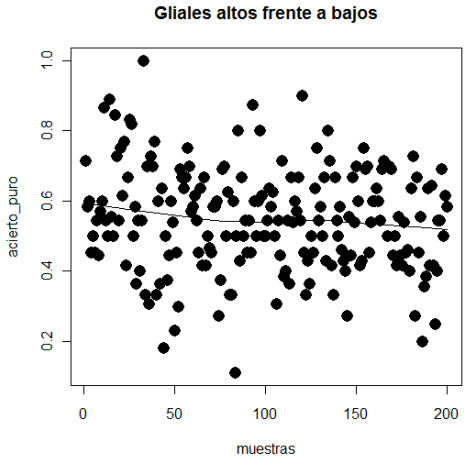
Clasificación de Gliales altos frente a Gliales bajos con 10 genes. Método KRSTT.

Numero total de casos de Gliales Bajos = 32

Los Gliales Bajos han sido codificados como CLASE1

Numero total de casos de Gliales Altos = 48

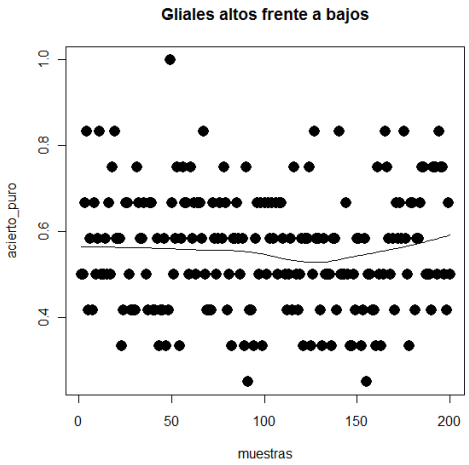
Los Gliales Altos han sido codificados como CLASE2



	1st Qu	Mean	3rd Qu
acierto_clase1	0.2500	0.4400	0.5000
acierto_clase2	0.5000	0.6124	0.7500
acierto	0.4444	0.5472	0.6380
acierto_marginal	0.4250	0.5262	0.6250

Tabla 23. Gliales altos frente a bajos KRSTT

Método Hold-one-out.



	1st Qu	Mean	3rd Qu
acierto_clase1	0.2500	0.4400	0.5000
acierto_clase2	0.5000	0.6125	0.7500
acierto	0.4167	0.5550	0.6667
acierto_marginal	0.4375	0.5262	0.6250

Tabla 24. Gliales altos frente a bajos Hold-one-out

Genes	213477_x_at	212284_x_at	201492_s_at	212869_x_at	1553588_at
Veces seleccionados	121	119	105	101	93
Genes	201429_s_at	207783_x_at	214327_x_at	AFFX-hum_alu_at	208834_x_at
Veces seleccionados	92	85	77	75	68

Tabla 25. Genes seleccionados un número mayor de veces con Hold-one-out

Clasificación de grados: GG2= (AS2,OA2,OD2) ,GG3= (AS3,OA3,OD3) y GBM con 10 genes. Método KRSTT.

Numero total de casos de Gliales Bajos = 32

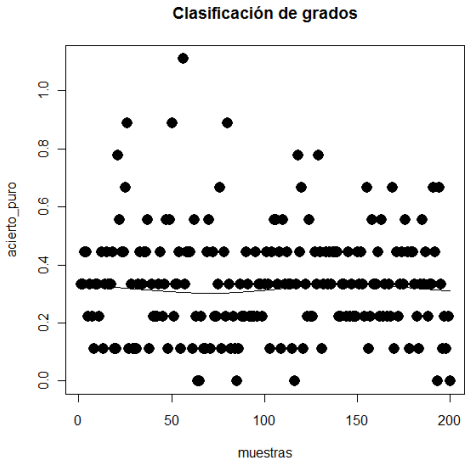
Los Gliales Bajos han sido codificados como CLASE1

Numero total de casos de Glioblastomas = 38

Los Glioblastomas han sido codificados como CLASE2

Numero total de casos de Gliales Anaplásticos =10

Los Gliales Anaplásticos han sido codificados como CLASE3



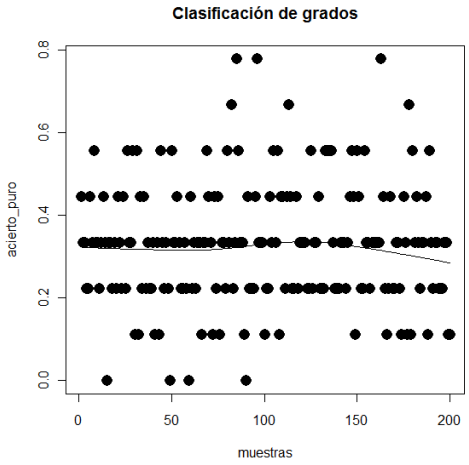
	1st Qu	Mean	3rd Qu
acierto_clase1	0.0000	0.2483	0.4000
acierto_clase2	0.2500	0.3688	0.5000
Acierto_clase3	0.000	0.112	0.200
acierto	0.2222	0.3089	0.4444
acierto_marginal	0.1667	0.2430	0.3278

Tabla 26. GG2 vs GG3 vs GBM KRSTT

Genes	213477_x_at	1553588_at	212284_x_at	201492_s_at	201429_s_at
Veces seleccionados	118	105	103	97	96
Genes	AFFX-hum_alu_at	211445_x_at	212869_x_at	53912_at	AFFX-r2-P1-cre-3_at
Veces seleccionados	90	82	78	77	76

Tabla 27. Genes seleccionados un número mayor de veces con KRSTT

Hold-one-out



	1st Qu	Mean	3rd Qu
acierto_clase1	0.0000	0.2467	0.4000
acierto_clase2	0.0000	0.292	0.5000
Acierto_clase3	0.0000	0.1115	0.2000
acierto	0.2222	0.3372	0.4444
acierto_marginal	0.1327	0.2167	0.3000

Tabla 28. GG2 vs GG3 vs GBM Hold-one-out

Genes	213477_x_at	1553588_at	212284_x_at	201492_s_at	201429_s_at
Veces seleccionados	121	105	105	98	93
Genes	212869_x_at	AFFX-hum_alu_at	207783_x_at	211445_x_at	214327_x_at
Veces seleccionados	85	79	77	76	71

Tabla 29. Genes seleccionados un número mayor de veces con Hold-one-out

Estudio del número de genes supervisados a utilizar

Hemos aplicado LDA varias veces a los datos, variando el número de genes seleccionados. Las diferentes cantidades de genes con las que hemos probado son 5, 10, 15, 20, 50 y 100.

	Min.	1st Qu	Median	Mean	3rd Qu	Max
Acierto	0.0000	0.3333	0.4286	0.4454	0.5714	1.0000
acierto_marginal	0.0000	0.2500	0.3958	0.4010	0.5000	0.8750

Tabla 30. Gliales bajos forzando a 5 genes significativos

	Min.	1st Qu	Median	Mean	3rd Qu	Max
Acierto	0.0000	0.3333	0.5000	0.4820	0.6000	1.0000
acierto_marginal	0.0000	0.2500	0.4167	0.4394	0.5833	1.0000

Tabla 31. Gliales bajos forzando a 10 genes significativos

	Min.	1st Qu	Median	Mean	3rd Qu	Max
Acierto	0.0000	0.3333	0.5000	0.4622	0.6062	1.0000
acierto_marginal	0.0000	0.2500	0.4250	0.4174	0.5417	1.0000

Tabla 32. Gliales bajos forzando a 15 genes significativos

	Min.	1st Qu	Median	Mean	3rd Qu	Max
Acierto	0.0000	0.3333	0.5000	0.4936	0.6354	1.0000
acierto_marginal	0.0000	0.2500	0.5000	0.4324	0.5437	1.0000

Tabla 33. Gliales bajos forzando a 20 genes significativos

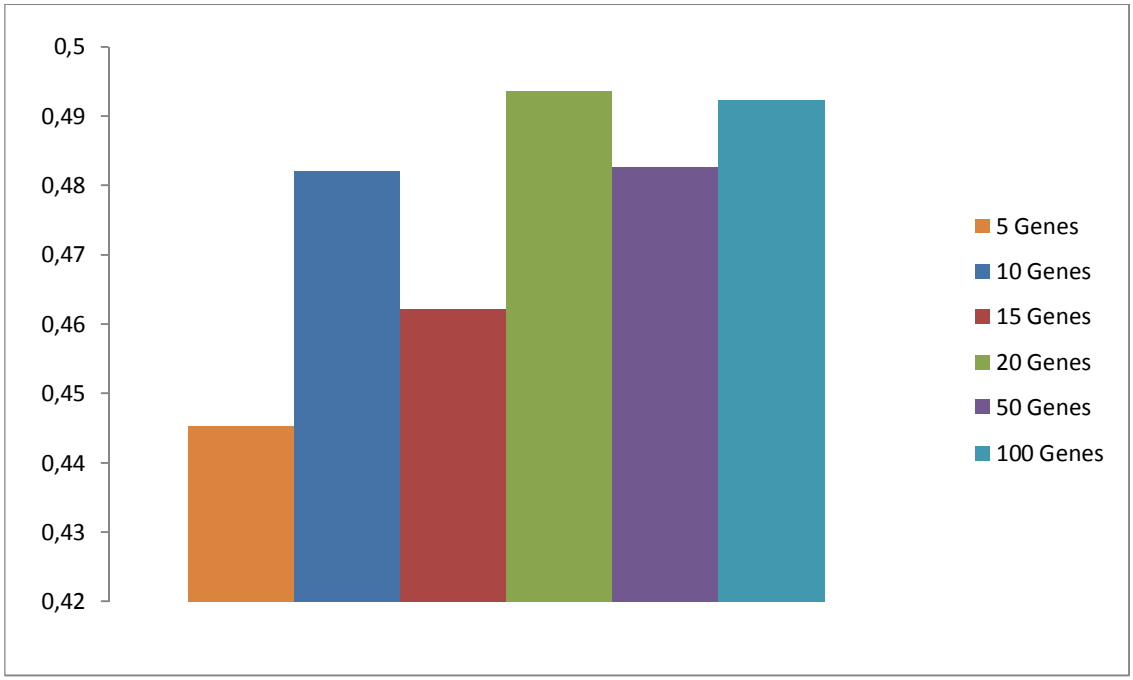
	Min.	1st Qu	Median	Mean	3rd Qu	Max
Acierto	0.0000	0.3750	0.5000	0.4827	0.6000	1.0000
acierto_marginal	0.0	0.3333	0.4583	0.4401	0.5521	1.0000

Tabla 34. Gliales bajos forzando a 50 genes significativos

	Min.	1st Qu	Median	Mean	3rd Qu	Max
Acierto	0.0000	0.3750	0.5000	0.4924	0.6000	1.0000
acierto_marginal	0.0000	0.3333	0.5000	0.4421	0.5417	0.8333

Tabla 35. Gliales bajos forzando a 100 genes significativos

En la gráfica 1 se nos muestra la tendencia de la tasa máxima de acierto conforme aumentamos el número de genes significativos que tiene que encontrar el clasificador.



Gráfica 1. Estudio LDA con diferente número de Genes significativos

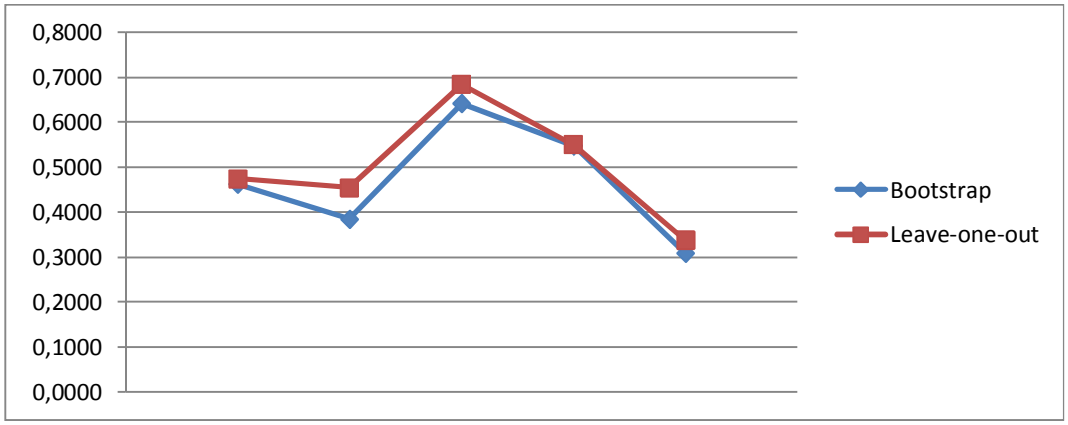
Selección del clasificador óptimo

Para la clasificación hemos utilizado dos métodos diferentes en la selección de variables: kRSTT y Hold-one-out. En función de los resultados podremos decir cual de los dos es más apropiado para nuestro conjunto de datos.

	Gliales bajos	Anaplásticos vs Glioblastomas	Glioblastomas vs resto	Altos vs bajos	Clasificación grados
KRSTT	0,4620	0,3844	0,6413	0,5472	0,3089
Hold-one-out	0,4742	0,4543	0,6839	0,5500	0,3372

Tabla 36. Comparación de los mejores resultados para los distintos problemas de diagnóstico

En el eje X tendremos como variables los diferentes experimentos expuestos en la tabla anterior en el mismo orden; en el eje Y la probabilidad de clasificar correctamente un gen, lo que hemos venido llamando "Acierto".



Gráfica 2. Comparación de los mejores resultados para los distintos problemas de diagnóstico

Elaboración de un Top10 genético

Desde el punto de vista de los genes seleccionados en cada comparación, hemos realizado el estudio de qué genes son seleccionados un número mayor de veces.

Para cada comparación y para cada método, nos hemos guardado los 10 genes más veces seleccionados, como podemos observar en la tabla 27:

Hold-one-out									
Gliales Bajos		Gliales Altos vs.Bajos		Anaplas. vs. Glioblas.		Glioblas. vs. resto		Clasificación Grados	
212284_x_at	110	213477_x_at	131	213477_x_at	199	213477_x_at	133	213477_x_at	135
212869_x_at	109	212284_x_at	123	201429_s_at	143	212284_x_at	127	1553588_at	124
211943_x_at	101	201492_s_at	120	1553588_at	130	207783_x_at	120	212284_x_at	117
AFFX-hum_alu_at	97	212869_x_at	108	211296_x_at	122	1553588_at	107	201492_s_at	102
211445_x_at	88	1553588_at	108	201492_s_at	119	212869_x_at	105	201429_s_at	96
207783_x_at	84	201429_s_at	106	213583_x_at	95	214327_x_at	92	212869_x_at	85
1553588_at	71	207783_x_at	99	204892_x_at	93	201492_s_at	80	AFFX-hum_alu_at	69
218601_at	67	214327_x_at	86	212284_x_at	82	208834_x_at	74	207783_x_at	66
200714_x_at	63	AFFX-hum_alu_at	75	208695_s_at	67	AFFX-hum_alu_at	74	211445_x_at	65
213477_x_at	61	208834_x_at	68	208825_x_at	64	211943_x_at	71	214327_x_at	59

Tabla 37. Top10 de genes seleccionados en los experimentos

Para asegurarnos de que los resultados son fiables, hemos realizado una prueba más. Vamos a repetir varias veces el experimento que mejor resultado nos ha dado, para poder observar que genes selecciona cada vez. Esperamos observar que selecciona los mismos genes, con pequeñas variaciones. Hemos repetido 4 veces el proceso para cada unos de ambos métodos de clasificación.

KRSTT			
213477_x_at	213477_x_at	213477_x_at	213477_x_at
212284_x_at	212284_x_at	212284_x_at	212284_x_at
207783_x_at	207783_x_at	207783_x_at	207783_x_at
1553588_at	1553588_at	1553588_at	1553588_at
212869_x_at	212869_x_at	212869_x_at	212869_x_at
214327_x_at	214327_x_at	214327_x_at	214327_x_at
201492_s_at	201492_s_at	213583_x_at	AFFX-hum_alu_at
213583_x_at	208834_x_at	201492_s_at	AFFX-r2-P1-cre-3_at
AFFX-hum_alu_at	AFFX-hum_alu_at	208834_x_at	211445_x_at
211943_x_at	211943_x_at	AFFX-hum_alu_at	53912_at

Hold-one-out			
213477_x_at	213477_x_at	213477_x_at	213477_x_at
212284_x_at	212284_x_at	212284_x_at	212284_x_at
207783_x_at	207783_x_at	207783_x_at	207783_x_at
1553588_at	1553588_at	1553588_at	1553588_at
212869_x_at	212869_x_at	212869_x_at	212869_x_at
214327_x_at	201492_s_at	214327_x_at	214327_x_at
201492_s_at	214327_x_at	208834_x_at	208834_x_at
208834_x_at	AFFX-hum_alu_at	208834_x_at	AFFX-hum_alu_at
AFFX-hum_alu_at	211943_x_at	AFFX-hum_alu_at	201492_s_at
211943_x_at	208834_x_at	201492_s_at	AFFX-r2-P1-cre-3_at

Zona sin cambios en la elección de genes

Zona con alteraciones en la elección de genes

Tabla 38. Genes elegidos de Glioblastomas frente al resto de clases tumorales

Conclusiones

Estudio del número de genes supervisados a utilizar

En primer lugar, hemos aplicado el proceso LDA varias veces a los datos pero con un número de genes significativos diferente cada vez. Observando la gráfica 1, podemos decir que a mayor número de genes significativos mayor es la tasa de acierto, lo que puede indicar una sobreestimación del acierto en los métodos complejos (con muchas variables) debido al método de remuestreo. Con 5 genes apenas son fiables los resultados debido a la poca cantidad de información.

En segundo lugar, tenemos las clasificaciones supervisadas de subtipos de Gliomas. Hemos realizado varias clasificaciones con diferentes mezclas de las clases tumorales con el fin de obtener unos resultados de calidad, calculando además estadísticos en los que podamos apoyarnos para emitir un juicio certero. En el mejor de los casos hemos llegado a un 68% de acierto para el experimento de Glioblastomas frente al resto de clases tumorales[8]. El resto de resultados son algo más pobres ya que la tasa de error aumenta.

Selección de genes diferenciados por expresión: el método óptimo

En la gráfica 2 podemos observar claramente como el método Hold-one-out, es decir el método basado en la validación cruzada que no utiliza repetición de las muestras, da mejores resultados en todos los experimentos realizados.

Elaboración de un Top10 genético

En la tabla 27 tenemos la clasificación de genes, de mayor número de repeticiones a menor. Podemos obserbar como en tonos marrones tenemos los genes que aparecen en todos o casi todos los experimentos, y en tonos azules los que tienen menos relevancia.

Además hemos podido comprobar que los datos ofrecidos tienen una calidad suficiente, ya que al repetir el mismo experiemnto una y otra vez, los genes resultates no varían, excepto los menos seleccionados. Para ello hemos repetido el experimento que mejores resultados ha dod en función de la mínima tasa de error. En la tabla 28 podemos observar que los genes 212284_x_at y 212869_x_at aparecen como seleccionados en todas las repeticiones del experimento y con un gran número de repeticiones.

Control de calidad de microarrays de expresión genética y Pre procesado de microarrays

Al aplicar a los datos el método JustRMA, esto normaliza los datos para suavizar y reducir los errores tanto técnicos como biológicos. De esta manera, forzará a las distribuciones de intensidad diferentes a igualarse, quedando identicas distribuciones en todas las muestras.

A la vista de los resultados[9] no hemos observado errores en el apartado de control de calidad, ni en el de pre procesado de microarrays.

[8] Véase la Tabla 26
[9] Véase la Gráfica 1 y las tablas 30 - 35

Bibliografía

1. **Affymetrix.** *Genechip Expression Analysis. Data Analysis Fundamentals.* 2002.

2. **Affymetrix**. *Microarray Suite 5.0 - User's Guide.* 2002.

3. **Genoma España.** *Microarrays y Biochips de ADN.* 2002.

4. *DNA Microarrays Data Analysis.* **Pasanen, T., y otros.** 2003, CSC - Scientific Computing Ltd.

5. *Exploration, Normalization and Summaries of High Density Oligonucleotide Array Probe Level Data.* **Irizarry, R. et al.** 2003, Biostadistics.

6. **Affymetrix.** [En línea] http://www.affymetrix.com.

7. **Bioconductor.** [En línea] http://www.bioconductor.org.

8. **cRAN.** R-project. [En línea] http://www.r-project.org.

9. **Robust Multiarray Average (RMA).**

   - Irizarry, R., Hobbs, B., et al. (2003) 'Exploration, normalization, and summaries of high density oligonucliotide array probe level data' . Biostatistics, in press.
   - Bolstad,B., Irizarry, R., Åstrand, M., and Speed, T. (2002) 'A comparison of normalization methods for high density oligonucleotide array data based on variance and bias'. Bioinformatics
   - Bolstad,B. (2001) 'Probe level quantile normalization of high density oligonucleotide array data'. http://www.stat.berkely.edu/~bolstad/

Anexos

Anexo 1. Código de Gliales altos frente a bajos.

```
##########################################################
###                                                    ###
###    Gliales bajos frente a altos                    ###
###                                                    ###
##########################################################
  library(limma)
  library(MASS)
  library(annotate)
  library(geneplotter)
  library(hgu133plus2.db)
  library(GOstats)
  library(Category)
  library(e1071)
  library(affy)
  library(affyPLM)
  library(simpleaffy)
  # Funcion que comprueba que el fichero exista
  check.files <- function(x) {
  if (file.exists(x)) {
  return(x)
  }
  else return(NULL)
  }
  # Aumentamos la memoria que puede utilizar el R a 4 Mb
  memory.size(max = 4000)
  ##########################################################
  ###    Funciones necesarias                            ###
  ##########################################################

  mapping <- function(labels,classesList,mappingList,default=NA,na.keep=TRUE)
  {
        if (is.na(default))
        {              res <- labels  }
        else    { res <- rep(default,length(labels))  }

        for (i in 1:length(classesList))
        {
               res[labels %in% mappingList[[i]]] <-  classesList[[i]]
        }
        if (na.keep)
        {       res[is.na(labels)]=NA  }
        return(res)
  }

  fusiona <- function (v1, v2) {
        for (i in 1: length(v1))
              if  (v1[i] == 'other' && v2[i] == 'other'){
                           res[i] = 'other'
            }
            else {
                  if (v1[i] != 'other') {
                       res[i] = v1[i]
               }
                  else {
                       res[i] = v2[i]
               }
           }
          return (res)
  }
  ##########################################################
  ###                                                    ###
  ###    Filtramos por etiquetas GG2, HGG                ###
  ###                                                    ###
  ##########################################################
  path.corpus <- paste(".", "corpus/BASEDATOS/BDDescom", sep = "/")
  samples <- "etDB_ma_090625.csv"
  myPhenoData <- read.AnnotatedDataFrame(samples, path.corpus,sep = ";", header = TRUE)
  labels <- myPhenoData$CR.CONSENSUS.CLINICAL.DIAGNOSIS
  #Elimina los espacios en blanco de las etiquetas
  labels <- gsub(" ", "", labels)
```

```
AS2 <- 'AS2'
AS2Set <- c('DIFFUSEASTROCYTOMA9400/3','FIBRILLARYASTROCYTOMA9420/3')
OA2 <- 'OA2'
OA2Set <- c('OLIGOASTROCYTOMA9382/3')
OD2 <- 'OD2'
OD2Set <- c('OLIGODENDROGLIOMA9450/3')
MEN <- 'MEN'
MENSet <-
c('MENINGIOMA9530/0','SECRETORYMENINGIOMA9530/0','FIBROUSMENINGIOMA9532/0','MENINGOTHELIALMENI
NGIOMA9531/0','TRANSITIONALMENINGIOMA9537/0')
GBM <- 'GBM'
GBMSet <- c('GLIOBLASTOMA9440/3','GIANTCELLGLIOBLASTOMA9441/3','GLIOSARCOMA9442/3')
MET <- 'MET'
METSet <- 'METASTASIS8000/6'
GG2 <- 'GG2'
GG2Set <- c(AS2Set,OA2Set,OD2Set)
EPE <- 'EPE'
EPESet <- c('EPENDYMOMA9391/3')
JPA <- 'JPA'
JPASet <- c('PILOCYTICASTROCYTOMA9421/1')
GG1 <- 'GG1'
GG1Set <- c(EPESet,JPASet)
AS3 <- 'AS3'
AS3Set <- c('ANAPLASTICASTROCYTOMA9401/3')
OA3 <- 'OA3'
OA3Set <- c('ANAPLASTICOLIGOASTROCYTOMA9382/3')
OD3 <- 'OD3'
OD3Set <- c('ANAPLASTICOLIGODENDROGLIOMA9451/3')
GG3 <- 'GG3'
GG3Set <- c(OA3Set,OD3Set,AS3Set)
LGG <- 'LGG'
LGGSet <- c(GG1Set,GG2Set)
HGG <- 'HGG'
HGGSet <- c(GG3Set,GBMSet)
MED <- 'MED'
MEDSet <- 'MEDULLOBLASTOMA9470/3'
LYM <- 'LYM'
LYMSet <- 'MALIGNANTLYMPHOMAS 9590/3'

### mapeo
HighLowGlialClasses <- list(GG2,HGG)
HighLowGlialList <- list(GG2Set,HGGSet)
HighLowGlialLabels <- mapping(labels,HighLowGlialClasses,HighLowGlialList,'other')
myPhenoData$HighLowGlialLabels <- HighLowGlialLabels

#########################################################
###                                                   ###
###    Creamos el exprdata                            ###
###                                                   ###
#########################################################
  myPhenoData <- myPhenoData[myPhenoData$HighLowGlialLabels %in% c("GG2","HGG")]
  if ((length(grep(".cel", myPhenoData$CASE.CODE)) == 0)){
  lfiles <- transform(pData(myPhenoData), CASE.CODE = paste(CASE.CODE,
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""))
  }
  lfiles <- paste(path.corpus, lfiles$CASE.CODE, sep = "/")
  lfileschecked <- sapply(lfiles, check.files)
  positions2keep <- which(!unlist(t(lapply(lfileschecked, is.null))))
  myPhenoData <- myPhenoData[positions2keep]

  Qname <- character()
  exprdata <- justRMA(filenames= paste(substr(path.corpus, 3, 60), "/",
myPhenoData$CASE.CODE[!(myPhenoData$CASE.CODE %in% Qname)],
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""),
  verbose=TRUE,
  sampleNames=myPhenoData$CASE.FILENAME[!(myPhenoData$CASE.CODE %in% Qname)],
  phenoData=myPhenoData)

#########################################################
###                                                   ###
###    Prepara Corpus                                 ###
###                                                   ###
```

```
#########################################################
### ETIQUETAS

labels2 <- phenoData(exprdata)$HighLowGlialLabels
design<- model.matrix(~factor(labels2))
colnames(design) <- c("GG2","HGG")
rownames(design) <- exprdata$CASE.CODE

### FUNCION ALEATORIO . La variable x es el conjunto de etiquetas .
### Devuelve las etiquetas separadas en dos grupos, uno para training y otro para test.
 funcionAl <- function (x)
 {
   trainingLabSamples <- list()
   testLabSamples <- list()
   traingPosSamples <- c()
   testPosSamples <- c()
   # pasamos el vector de etiquetas de CHAR a factor
   xfactor <- factor(x)

   for(j in 1:length(levels(xfactor)))
   {
     posLab <- which(xfactor == levels(xfactor)[j])
     #  Solo nos queremos quedar con unas 10 variables más o menos

     #-------------------------------------
     #    C O N     R E P E T I C I O N   |
     #-------------------------------------
     if (length(posLab) > 10)
        permutacion <- sample(posLab,round(length(posLab)*0.5) ,replace = TRUE)
     else
        permutacion <- sample(posLab,round(length(posLab)) ,replace = TRUE)
     trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.6)]; # Separamos la
parte de Training de la de Test
     traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     testLabSamples[[j]] <- setdiff(permutacion, trainingLabSamples[[j]])

     #-------------------------------------
     #    S I N     R E P E T I C I O N   |
     #-------------------------------------
     #permutacion <- sample(posLab,round(length(posLab)*0.5))
     #trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.7)];
     #traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     #testLabSamples[[j]] <-
permutacion[(round(length(permutacion)*0.7)+1):(length(permutacion))]

     testPosSamples <- c(testPosSamples, testLabSamples[[j]])
   }
   list(traingPosSamples = traingPosSamples, testPosSamples = testPosSamples)
 }
 # Cuantos genes significativos queremos que sean seleccionados
 geneselecionados <- 10
 genesPos <- list()
 diffExpressedGenesLista <- list()
 n_test <- 9
 Acierto <- vector()
 acierto_clase1 <- vector()
 acierto_clase2 <- vector()
 acierto_clase3 <- vector()
 acierto_marginal <- vector()
for (i in 1:200)
{
       print.default(i)

  rm(labels2)
  rm (design)

  labels2 <- phenoData(exprdata)$HighLowGlialLabels
  # nos guardamos las etiquetas
  design<- model.matrix(~factor(labels2))
  summary(labels2)
  colnames(design) <- c("GG2","HGG")
  rownames(design) <- exprdata$CASE.CODE
  muestraTraingTest<- funcionAl(phenoData(exprdata)$HighLowGlialLabels)
  exprDataTraining <- exprdata[,muestraTraingTest$traingPosSamples]
  ###diseño de matriz (solo con training)
```

```
  design <- model.matrix(~factor(phenoData(exprDataTraining)$HighLowGlialLabels))
  ### lmfit (solo con training)
  eb <- eBayes(lmFit(exprs(exprDataTraining), design))
  difExpressedGenesTable <- topTable(eb, number = geneselecionados, adjust = "fdr") #Numero de
genes que cogemos <--
  rm(eb)
  diffExpressedGenesLista[[i]] <- difExpressedGenesTable[difExpressedGenesTable[, 6] <1e-04,
1]
  genesPos[[i]] <- which(featureNames(exprDataTraining) %in% diffExpressedGenesLista[[i]])
  ### lda (solo con training)
  ### Comprobamos y limpiamos los posibles valores a 0
  for( j in 1:length(diffExpressedGenesLista))
   {
      if (length(diffExpressedGenesLista[[j]]) <= 0)
          diffExpressedGenesLista[[j]] <- "Void"
   }

  z <- lda(factor(phenoData(exprDataTraining)$HighLowGlialLabels) ~
.,data.frame(t(exprs(exprDataTraining[(diffExpressedGenesLista[[i]]), ])),row.names = NULL))
  exprDataTest <- exprdata[,muestraTraingTest$testPosSamples]
  TestResult<- predict(z,
data.frame(t(exprs(exprDataTest[diffExpressedGenesLista[[i]],])))$class
  labels <- factor(labels2)
  # cambiar
  TestResult <- factor(TestResult)
  levels(TestResult) <- levels(factor(labels2))
  confusMatrix <- table(labels2[1:length(TestResult)], TestResult)
  #----------------------------
  if (!levels(TestResult)[1] %in% TestResult) {
        confusMatrix <- rbind(c(0,0), confusMatrix)
        acierto_clase1[[i]]   <- 0
        acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
  }
  if((!levels(TestResult)[2] %in% TestResult)||(length(TestResult)<=4)){
        confusMatrix <- rbind(confusMatrix, c(0,0))
        acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
        acierto_clase2[[i]]   <- 0
  }
  if((levels(TestResult)[1] %in% TestResult)&&(levels(TestResult)[2] %in%
TestResult)&&(length(TestResult)>4)){
        acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
           acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
  }
  n_test <- confusMatrix[[1,1]]+confusMatrix[[2,1]]+confusMatrix[[1,2]]+confusMatrix[[2,2]]
  Acierto[[i]]  <- (confusMatrix[[1,1]]+confusMatrix[[2,2]])/n_test
  acierto_marginal[[i]] <- (acierto_clase1[[i]]+acierto_clase2[[i]])/2
}
```

Anexo 2. Código de Gliales bajos.

```
##########################################################
###                                                    ###
###    AS2 vs. OA2 vs.OD2                              ###
###                                                    ###
##########################################################
  library(limma)
  library(MASS)
  library(annotate)
  library(geneplotter)
  library(hgu133plus2.db)
  library(GOstats)
  library(Category)
  library(e1071)
  library(affy)
  library(affyPLM)
  library(simpleaffy)
  # Función que comprueba que el fichero exista
  check.files <- function(x) {
  if (file.exists(x)) {
  return(x)
  }
```

```
  else return(NULL)
  }
  # Aumentamos la memoria que puede utilizar el R a 4 Mb
  memory.size(max = 4000)
  ############################################################
  ###    Funciones necesarias                              ###
  ############################################################

  mapping <- function(labels,classesList,mappingList,default=NA,na.keep=TRUE)
  {
        if (is.na(default))
        {              res <- labels  }
        else    { res <- rep(default,length(labels))  }

        for (i in 1:length(classesList))
        {
               res[labels %in% mappingList[[i]]] <-  classesList[[i]]
        }
        if (na.keep)
        {       res[is.na(labels)]=NA  }
        return(res)
  }

  fusiona <- function (v1, v2) {
        for (i in 1: length(v1))
              if  (v1[i] == 'other' && v2[i] == 'other'){
                           res[i] = 'other'
            }
            else {
                  if (v1[i] != 'other') {
                       res[i] = v1[i]
               }
                  else {
                       res[i] = v2[i]
               }
           }
          return (res)
  }
  ############################################################
  ###                                                      ###
  ###    Filtramos por etiquetas AS2, OLIGO                ###
  ###                                                      ###
  ############################################################
  path.corpus <- paste(".", "corpus/BASEDATOS/BDDescom", sep = "/")
  samples <- "etDB_ma_090625.csv"
  myPhenoData <- read.AnnotatedDataFrame(samples, path.corpus,sep = ";", header = TRUE)
  labels <- myPhenoData$CR.CONSENSUS.CLINICAL.DIAGNOSIS
  #Elimina los espacios en blanco de las etiquetas
  labels <- gsub(" ", "", labels)

AS2 <- 'AS2'
AS2Set <- c('DIFFUSEASTROCYTOMA9400/3','FIBRILLARYASTROCYTOMA9420/3')
OA2 <- 'OA2'
OA2Set <- c('OLIGOASTROCYTOMA9382/3')
OD2 <- 'OD2'
OD2Set <- c('OLIGODENDROGLIOMA9450/3')
Oligo <- 'OLIGO'
OligoSet <- c(OA2Set,OD2Set)
MEN <- 'MEN'
MENSet <-
c('MENINGIOMA9530/0','SECRETORYMENINGIOMA9530/0','FIBROUSMENINGIOMA9532/0','MENINGOTHELIALMENI
NGIOMA9531/0','TRANSITIONALMENINGIOMA9537/0')
GBM <- 'GBM'
GBMSet <- c('GLIOBLASTOMA9440/3','GIANTCELLGLIOBLASTOMA9441/3','GLIOSARCOMA9442/3')
MET <- 'MET'
METSet <- 'METASTASIS8000/6'
GG2 <- 'GG2'
GG2Set <- c(AS2Set,OA2Set,OD2Set)
EPE <- 'EPE'
EPESet <- c('EPENDYMOMA9391/3')
JPA <- 'JPA'
JPASet <- c('PILOCYTICASTROCYTOMA9421/1')
GG1 <- 'GG1'
GG1Set <- c(EPESet,JPASet)
AS3 <- 'AS3'
```

```
AS3Set <- c('ANAPLASTICASTROCYTOMA9401/3')
OA3 <- 'OA3'
OA3Set <- c('ANAPLASTICOLIGOASTROCYTOMA9382/3')
OD3 <- 'OD3'
OD3Set <- c('ANAPLASTICOLIGODENDROGLIOMA9451/3')
GG3 <- 'GG3'
GG3Set <- c(OA3Set,OD3Set,AS3Set)
LGG <- 'LGG'
LGGSet <- c(GG1Set,GG2Set)
HGG <- 'HGG'
HGGSet <- c(GG3Set,GBMSet)
MED <- 'MED'
MEDSet <- 'MEDULLOBLASTOMA9470/3'
LYM <- 'LYM'
LYMSet <- 'MALIGNANTLYMPHOMAS 9590/3'

### mapeo
LowGlialClasses <- list(AS2,Oligo)
LowGlialList <- list(AS2Set,OligoSet)
LowGlialLabels <- mapping(labels,LowGlialClasses,LowGlialList,'other')
myPhenoData$LowGlialLabels <- LowGlialLabels

##########################################################
###                                                    ###
###    Creamos el exprdata                             ###
###                                                    ###
##########################################################
  myPhenoData <- myPhenoData[myPhenoData$LowGlialLabels %in% c("AS2","OLIGO")]
  if ((length(grep(".cel", myPhenoData$CASE.CODE)) == 0))
    #  &&(myPhenoData$LowGlialLabels %in% c("AS2","OA2","OD2")))
  {
  lfiles <- transform(pData(myPhenoData), CASE.CODE = paste(CASE.CODE,
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""))
  }
  lfiles <- paste(path.corpus, lfiles$CASE.CODE, sep = "/")#[myPhenoData$LowGlialLabels %in%
c("AS2","OA2","OD2")]
  lfileschecked <- sapply(lfiles, check.files)
  positions2keep <- which(!unlist(t(lapply(lfileschecked, is.null))))
  myPhenoData <- myPhenoData[positions2keep]

  Qname <- character()
  exprdata <- justRMA(filenames= paste(substr(path.corpus, 3, 60), "/",
myPhenoData$CASE.CODE[!(myPhenoData$CASE.CODE %in% Qname)],
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""),
  verbose=TRUE,
  sampleNames=myPhenoData$CASE.FILENAME[!(myPhenoData$CASE.CODE %in% Qname)],
  phenoData=myPhenoData)

##########################################################
###                                                    ###
###    Prepara Corpus                                  ###
###                                                    ###
##########################################################
### ETIQUETAS

labels2 <- phenoData(exprdata)$LowGlialLabels
design<- model.matrix(~factor(labels2))

colnames(design) <- c("AS2", "OLIGO")
rownames(design) <- exprdata$CASE.CODE

### FUNCION FIT

 funcionFit <- function (x,y)
  {
      return (eBayes(lmFit(x, y)))
  }

### FUNCION ALEATORIO . La variable x es el conjunto de etiquetas .
### Devuelve las etiquetas separadas en dos grupos, uno para training y otro para test.

 funcionAl <- function (x)
 {
```

```
   trainingLabSamples <- list()
   testLabSamples <- list()
   traingPosSamples <- c()
   testPosSamples <- c()
   # pasamos el vector de etiquetas de CHAR a factor
   xfactor <- factor(x)
   for(j in 1:length(levels(xfactor)))
   {
     posLab <- which(xfactor == levels(xfactor)[j])
     #  Solo nos queremos quedar con unas 10 variables más o menos

     #-----------------------------------
     #    C O N     R E P E T I C I O N   |
     #-----------------------------------
     permutacion <- sample(posLab,length(posLab) ,replace = TRUE)
     trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.6)];
     # Separamos la parte de Training de la de Test
     traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     testLabSamples[[j]] <- setdiff(permutacion, trainingLabSamples[[j]])

     #-----------------------------------
     #    S I N     R E P E T I C I O N   |
     #-----------------------------------
     #permutacion <- sample(posLab,round(length(posLab)*0.5))
     #trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.7)];
     #traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     #testLabSamples[[j]] <-
permutacion[(round(length(permutacion)*0.7)+1):(length(permutacion))]

     testPosSamples <- c(testPosSamples, testLabSamples[[j]])
   }
   list(traingPosSamples = traingPosSamples, testPosSamples = testPosSamples)
 }

 #-----------------------------------
 #    GRAN BUCLE   200 rep            |
 #-----------------------------------
 geneselecionados <- 10
 genesPos <- list()
 diffExpressedGenesLista <- list()
 n_test <- 9
 Acierto <- vector()
 acierto_clase1 <- vector()
 acierto_clase2 <- vector()
 acierto_clase3 <- vector()
 acierto_marginal <- vector()
for (i in 1:200)
{
       print.default(i)
  # Borramos estas estiquetas para reutilizarlas
  rm(labels2)
  rm (design)
  labels2 <- phenoData(exprdata)$LowGlialLabels
  design<- model.matrix(~factor(labels2))
  summary(labels2)

  colnames(design) <- c("AS2", "OLIGO")
  rownames(design) <- exprdata$CASE.CODE

  muestraTraingTest<- funcionAl(phenoData(exprdata)$LowGlialLabels)
  exprDataTraining <- exprdata[,muestraTraingTest$traingPosSamples]
  ###diseño de matriz (solo con training)
  design <- model.matrix(~factor(phenoData(exprDataTraining)$LowGlialLabels))
  eb <- eBayes(lmFit(exprs(exprDataTraining), design))

  difExpressedGenesTable <- topTable(eb, number = geneselecionados, adjust = "fdr") #Numero de
genes que cogemos <--
  rm(eb)
  diffExpressedGenesLista[[i]] <- difExpressedGenesTable[difExpressedGenesTable[, 6] <1e-04,
1]

  ### lda (solo con training)
  ### Comprobamos y limpiamos los posibles valores a 0
  for( j in 1:length(diffExpressedGenesLista))
   {
```

```
      if (length(diffExpressedGenesLista[[j]]) <= 0)
          diffExpressedGenesLista[[j]] <- "Void"
   }
  genesPos[[i]] <- which(featureNames(exprDataTraining) %in% diffExpressedGenesLista[[i]])
  z <- lda(factor(phenoData(exprDataTraining)$LowGlialLabels) ~
.,data.frame(t(exprs(exprDataTraining[(diffExpressedGenesLista[[i]]), ])),row.names = NULL))

  ##z <- lda(factor(labels2) ~
.,data.frame((exprs(exprDataTraining[(diffExpressedGenesLista[[1]]), ]))))
  exprDataTest <- exprdata[,muestraTraingTest$testPosSamples]
  TestResult<- predict(z,
data.frame(t(exprs(exprDataTest[diffExpressedGenesLista[[i]],])))$class


  labels <- factor(labels2)
  # cambiar
  TestResult <- factor(TestResult)
  levels(TestResult) <- levels(factor(labels2))
  confusMatrix <- table(labels2[1:length(TestResult)], TestResult)

     if (!levels(TestResult)[1] %in% TestResult) {
        confusMatrix <- rbind(c(0,0), confusMatrix)
        acierto_clase1[[i]]   <- 0
        acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
     }
     if((!levels(TestResult)[2] %in% TestResult)||(length(TestResult)<=4)){
        confusMatrix <- rbind(confusMatrix, c(0,0))
        acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
        acierto_clase2[[i]]   <- 0
     }
     if((levels(TestResult)[1] %in% TestResult)&&(levels(TestResult)[2] %in%
TestResult)&&(length(TestResult)>4)){
        acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
           acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
     }
  n_test <- confusMatrix[[1,1]]+confusMatrix[[2,1]]+confusMatrix[[1,2]]+confusMatrix[[2,2]]
  Acierto[[i]]  <- (confusMatrix[[1,1]]+confusMatrix[[2,2]])/n_test
  #acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
  #acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
  #acierto_clase3[[i]]   <-
(confusMatrix[3,3]/(confusMatrix[3,1]+confusMatrix[3,2]+confusMatrix[3,3]))
  acierto_marginal[[i]] <- (acierto_clase1[[i]]+acierto_clase2[[i]])/2

}
```

Anexo 3. Código de Gliales Anaplásticos frente a Glioblastomas.

```
#########################################################
###                                                   ###
###    Gliales anaplásticos frente a Glioblastomas    ###
###                                                   ###
#########################################################
  library(limma)
  library(MASS)
  library(annotate)
  library(geneplotter)
  library(hgu133plus2.db)
  library(GOstats)
  library(Category)
  library(e1071)
  library(affy)
  library(affyPLM)
  library(simpleaffy)
  # Funcion que comprueba que el fichero exista
  check.files <- function(x) {
  if (file.exists(x)) {
  return(x)
  }
  else return(NULL)
  }
  # Aumentamos la memoria que puede utilizar el R a 4 Mb
  memory.size(max = 4000)
  #########################################################
```

```
  ###    Funciones necesarias                                ###
  ############################################################

  mapping <- function(labels,classesList,mappingList,default=NA,na.keep=TRUE)
  {
        if (is.na(default))
        {               res <- labels  }
        else    { res <- rep(default,length(labels))  }

        for (i in 1:length(classesList))
        {
               res[labels %in% mappingList[[i]]] <-  classesList[[i]]
        }
        if (na.keep)
        {       res[is.na(labels)]=NA  }
        return(res)
  }

  fusiona <- function (v1, v2) {
        for (i in 1: length(v1))
              if  (v1[i] == 'other' && v2[i] == 'other'){
                           res[i] = 'other'
            }
            else {
                  if (v1[i] != 'other') {
                       res[i] = v1[i]
                }
                   else {
                       res[i] = v2[i]
                }
             }
           return (res)
  }
  ############################################################
  ###                                                      ###
  ###    Filtramos por etiquetas GBM, GG3                  ###
  ###                                                      ###
  ############################################################
  path.corpus <- paste(".", "corpus/BASEDATOS/BDDescom", sep = "/")
  samples <- "etDB_ma_090625.csv"
  myPhenoData <- read.AnnotatedDataFrame(samples, path.corpus,sep = ";", header = TRUE)
  labels <- myPhenoData$CR.CONSENSUS.CLINICAL.DIAGNOSIS
  #Elimina los espacios en blanco de las etiquetas
  labels <- gsub(" ", "", labels)

AS2 <- 'AS2'
AS2Set <- c('DIFFUSEASTROCYTOMA9400/3','FIBRILLARYASTROCYTOMA9420/3')
OA2 <- 'OA2'
OA2Set <- c('OLIGOASTROCYTOMA9382/3')
OD2 <- 'OD2'
OD2Set <- c('OLIGODENDROGLIOMA9450/3')
MEN <- 'MEN'
MENSet <-
c('MENINGIOMA9530/0','SECRETORYMENINGIOMA9530/0','FIBROUSMENINGIOMA9532/0','MENINGOTHELIALMENI
NGIOMA9531/0','TRANSITIONALMENINGIOMA9537/0')
GBM <- 'GBM'
GBMSet <- c('GLIOBLASTOMA9440/3','GIANTCELLGLIOBLASTOMA9441/3','GLIOSARCOMA9442/3')
MET <- 'MET'
METSet <- 'METASTASIS8000/6'
GG2 <- 'GG2'
GG2Set <- c(AS2Set,OA2Set,OD2Set)
EPE <- 'EPE'
EPESet <- c('EPENDYMOMA9391/3')
JPA <- 'JPA'
JPASet <- c('PILOCYTICASTROCYTOMA9421/1')
GG1 <- 'GG1'
GG1Set <- c(EPESet,JPASet)
AS3 <- 'AS3'
AS3Set <- c('ANAPLASTICASTROCYTOMA9401/3')
OA3 <- 'OA3'
OA3Set <- c('ANAPLASTICOLIGOASTROCYTOMA9382/3')
OD3 <- 'OD3'
OD3Set <- c('ANAPLASTICOLIGODENDROGLIOMA9451/3')
GG3 <- 'GG3'
GG3Set <- c(OA3Set,OD3Set,AS3Set)
```

```
LGG <- 'LGG'
LGGSet <- c(GG1Set,GG2Set)
HGG <- 'HGG'
HGGSet <- c(GG3Set,GBMSet)
MED <- 'MED'
MEDSet <- 'MEDULLOBLASTOMA9470/3'
LYM <- 'LYM'
LYMSet <- 'MALIGNANTLYMPHOMAS 9590/3'
### mapeo
AnaGlialClasses <- list(GBM,GG3)
AnaGlialList <- list(GBMSet,GG3Set)
AnaGlialLabels <- mapping(labels,AnaGlialClasses,AnaGlialList,'other')
myPhenoData$AnaGlialLabels <- AnaGlialLabels

#########################################################
###                                                   ###
###    Creamos el exprdata                            ###
###                                                   ###
#########################################################
  myPhenoData <- myPhenoData[myPhenoData$AnaGlialLabels %in% c("GBM","GG3")]
  if ((length(grep(".cel", myPhenoData$CASE.CODE)) == 0)){
  lfiles <- transform(pData(myPhenoData), CASE.CODE = paste(CASE.CODE,
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""))
  }
  lfiles <- paste(path.corpus, lfiles$CASE.CODE, sep = "/")#[myPhenoData$AnaGlialLabels %in%
c("AS2","OA2","OD2")]
  lfileschecked <- sapply(lfiles, check.files)
  positions2keep <- which(!unlist(t(lapply(lfileschecked, is.null))))
  myPhenoData <- myPhenoData[positions2keep]

  Qname <- character()
  exprdata <- justRMA(filenames= paste(substr(path.corpus, 3, 60), "/",
myPhenoData$CASE.CODE[!(myPhenoData$CASE.CODE %in% Qname)],
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""),
  verbose=TRUE,
  sampleNames=myPhenoData$CASE.FILENAME[!(myPhenoData$CASE.CODE %in% Qname)],
  phenoData=myPhenoData)

#########################################################
###                                                   ###
###    Prepara Corpus                                 ###
###                                                   ###
#########################################################
### ETIQUETAS

labels2 <- phenoData(exprdata)$AnaGlialLabels
design<- model.matrix(~factor(labels2))
colnames(design) <- c("GBM","GG3")
rownames(design) <- exprdata$CASE.CODE

### FUNCION ALEATORIO . La variable x es el conjunto de etiquetas .
### Devuelve las etiquetas separadas en dos grupos, uno para training y otro para test.

 funcionAl <- function (x)

 {
   trainingLabSamples <- list()
   testLabSamples <- list()
   traingPosSamples <- c()
   testPosSamples <- c()
   # pasamos el vector de etiquetas de CHAR a factor
   xfactor <- factor(x)

   for(j in 1:length(levels(xfactor)))
   {
     posLab <- which(xfactor == levels(xfactor)[j])
     #  Solo nos queremos quedar con unas 10 variables más o menos

     #-----------------------------------
     #    C O N     R E P E T I C I O N   |
     #-----------------------------------
     if (length(posLab) > 10)
        permutacion <- sample(posLab,round(length(posLab)*0.5) ,replace = TRUE)
```

```
     else
        permutacion <- sample(posLab,round(length(posLab)) ,replace = TRUE)
     trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.6)]; # Separamos la
parte de Training de la de Test
     traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     testLabSamples[[j]] <- setdiff(permutacion, trainingLabSamples[[j]])

     #-----------------------------------
     #    S I N     R E P E T I C I O N   |
     #-----------------------------------
     #permutacion <- sample(posLab,round(length(posLab)*0.5))
     #trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.7)];
     #traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     #testLabSamples[[j]] <-
permutacion[(round(length(permutacion)*0.7)+1):(length(permutacion))]

     testPosSamples <- c(testPosSamples, testLabSamples[[j]])
   }
   list(traingPosSamples = traingPosSamples, testPosSamples = testPosSamples)
 }

 geneselecionados <- 10 # Cuantos genes significativos queremos que sean seleccionados
 genesPos <- list()
 diffExpressedGenesLista <- list()
 n_test <- 9
 Acierto <- vector()
 acierto_clase1 <- vector()
 acierto_clase2 <- vector()
 acierto_clase3 <- vector()
 acierto_marginal <- vector()
for (i in 1:200)
{
       print.default(i)

  rm(labels2)
  rm (design)

  labels2 <- phenoData(exprdata)$AnaGlialLabels
  # nos guardamos las etiquetas
  design<- model.matrix(~factor(labels2))
  summary(labels2)
  colnames(design) <- c("GBM","GG3")
  rownames(design) <- exprdata$CASE.CODE
  muestraTraingTest<- funcionAl(phenoData(exprdata)$AnaGlialLabels)
  exprDataTraining <- exprdata[,muestraTraingTest$traingPosSamples]
  ###diseño de matriz (solo con training)
  design <- model.matrix(~factor(phenoData(exprDataTraining)$AnaGlialLabels))
  ### lmfit (solo con training)
  eb <- eBayes(lmFit(exprs(exprDataTraining), design))
  difExpressedGenesTable <- topTable(eb, number = geneselecionados, adjust = "fdr")
  rm(eb)
  diffExpressedGenesLista[[i]] <- difExpressedGenesTable[difExpressedGenesTable[, 6] <1e-04,
1]
  genesPos[[i]] <- which(featureNames(exprDataTraining) %in% diffExpressedGenesLista[[i]])
  ### lda (solo con training)
  ### Comprobamos y limpiamos los posibles valores a 0
  for( j in 1:length(diffExpressedGenesLista))
   {
      if (length(diffExpressedGenesLista[[j]]) <= 0)
          diffExpressedGenesLista[[j]] <- "Void"
   }

  z <- lda(factor(phenoData(exprDataTraining)$AnaGlialLabels) ~
.,data.frame(t(exprs(exprDataTraining[(diffExpressedGenesLista[[i]]), ])),row.names = NULL))

  ##z <- lda(factor(labels2) ~
.,data.frame((exprs(exprDataTraining[(diffExpressedGenesLista[[1]]), ]))))
  exprDataTest <- exprdata[,muestraTraingTest$testPosSamples]
  TestResult<- predict(z,
data.frame(t(exprs(exprDataTest[diffExpressedGenesLista[[i]],])))$class
  labels <- factor(labels2)
  TestResult <- factor(TestResult)
  levels(TestResult) <- levels(factor(labels2))
  confusMatrix <- table(labels2[1:length(TestResult)], TestResult)
  #----------------------------
```

```
  if (!levels(TestResult)[1] %in% TestResult) {
        confusMatrix <- rbind(c(0,0), confusMatrix)
        acierto_clase1[[i]]   <- 0
        acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
  }
  if((!levels(TestResult)[2] %in% TestResult)||(length(TestResult)<=4)){
        confusMatrix <- rbind(confusMatrix, c(0,0))
        acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
        acierto_clase2[[i]]   <- 0
  }
  if((levels(TestResult)[1] %in% TestResult)&&(levels(TestResult)[2] %in%
TestResult)&&(length(TestResult)>4)){
        acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
           acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
  }
  n_test <- confusMatrix[[1,1]]+confusMatrix[[2,1]]+confusMatrix[[1,2]]+confusMatrix[[2,2]]
  Acierto[[i]]  <- (confusMatrix[[1,1]]+confusMatrix[[2,2]])/n_test
  acierto_marginal[[i]] <- (acierto_clase1[[i]]+acierto_clase2[[i]])/2
}
```

Anexo 4. Código de Glioblastomas frente el resto de tumores.

```
############################################################
###                                                      ###
###    Glioblastomas frente el resto de tumores          ###
###                                                      ###
############################################################
  # Librerias necesarias
  library(limma)
  library(MASS)
  library(annotate)
  library(geneplotter)
  library(hgu133plus2.db)
  library(GOstats)
  library(Category)
  library(e1071)
  library(affy)
  library(affyPLM)
  library(simpleaffy)
  # Funcion que comprueba que el fichero exista
  check.files <- function(x) {
  if (file.exists(x)) {
  return(x)
  }
  else return(NULL)
  }
  # Aumentamos la memoria que puede utilizar el R a 4 Mb
  memory.size(max = 4000)
  ############################################################
  ###    Funciones necesarias                              ###
  ############################################################

  mapping <- function(labels,classesList,mappingList,default=NA,na.keep=TRUE)
  {
        if (is.na(default))
        {             res <- labels  }
        else    { res <- rep(default,length(labels))  }

        for (i in 1:length(classesList))
        {
               res[labels %in% mappingList[[i]]] <-  classesList[[i]]
        }
        if (na.keep)
        {       res[is.na(labels)]=NA  }
        return(res)
  }

  fusiona <- function (v1, v2) {
        for (i in 1: length(v1))
              if  (v1[i] == 'other' && v2[i] == 'other'){
                           res[i] = 'other'
            }
            else {
                  if (v1[i] != 'other') {
```

```
                    res[i] = v1[i]
                }
                   else {
                    res[i] = v2[i]
                }
            }
           return (res)
  }

  ##########################################################
  ###                                                    ###
  ###    Filtramos por etiquetas GBM y OTHER             ###
  ###                                                    ###
  ##########################################################
  path.corpus <- paste(".", "corpus/BASEDATOS/BDDescom", sep = "/")
  samples <- "etDB_ma_090625.csv"
  myPhenoData <- read.AnnotatedDataFrame(samples, path.corpus,sep = ";", header = TRUE)
  labels <- myPhenoData$CR.CONSENSUS.CLINICAL.DIAGNOSIS
  #Elimina los espacios en blanco de las etiquetas
  labels <- gsub(" ", "", labels)

AS2 <- 'AS2'
AS2Set <- c('DIFFUSEASTROCYTOMA9400/3','FIBRILLARYASTROCYTOMA9420/3')
OA2 <- 'OA2'
OA2Set <- c('OLIGOASTROCYTOMA9382/3')
OD2 <- 'OD2'
OD2Set <- c('OLIGODENDROGLIOMA9450/3')
MEN <- 'MEN'
MENSet <-
c('MENINGIOMA9530/0','SECRETORYMENINGIOMA9530/0','FIBROUSMENINGIOMA9532/0','MENINGOTHELIALMENI
NGIOMA9531/0','TRANSITIONALMENINGIOMA9537/0')
GBM <- 'GBM'
GBMSet <- c('GLIOBLASTOMA9440/3','GIANTCELLGLIOBLASTOMA9441/3','GLIOSARCOMA9442/3')
MET <- 'MET'
METSet <- 'METASTASIS8000/6'
GG2 <- 'GG2'
GG2Set <- c(AS2Set,OA2Set,OD2Set)
EPE <- 'EPE'
EPESet <- c('EPENDYMOMA9391/3')
JPA <- 'JPA'
JPASet <- c('PILOCYTICASTROCYTOMA9421/1')
GG1 <- 'GG1'
GG1Set <- c(EPESet,JPASet)
AS3 <- 'AS3'
AS3Set <- c('ANAPLASTICASTROCYTOMA9401/3')
OA3 <- 'OA3'
OA3Set <- c('ANAPLASTICOLIGOASTROCYTOMA9382/3')
OD3 <- 'OD3'
OD3Set <- c('ANAPLASTICOLIGODENDROGLIOMA9451/3')
GG3 <- 'GG3'
GG3Set <- c(OA3Set,OD3Set,AS3Set)
LGG <- 'LGG'
LGGSet <- c(GG1Set,GG2Set)
HGG <- 'HGG'
HGGSet <- c(GG3Set,GBMSet)
MED <- 'MED'
MEDSet <- 'MEDULLOBLASTOMA9470/3'
LYM <- 'LYM'
LYMSet <- 'MALIGNANTLYMPHOMAS 9590/3'

OTHER <- 'OTHER'
OTHERSet <- c(LGGSet, GG3Set, MEDSet, LYMSet)


### MAPEO
GlioClasses <- list(GBM,OTHER)
GlioList <- list(OTHERSet,GBMSet)
GlioLabels <- mapping(labels,GlioClasses,GlioList,'other')
myPhenoData$GlioLabels <- GlioLabels

##########################################################
###                                                    ###
###    Creamos el exprdata                             ###
###                                                    ###
##########################################################
```

```
  myPhenoData <- myPhenoData[myPhenoData$GlioLabels %in% c("GBM","OTHER")]
  if ((length(grep(".cel", myPhenoData$CASE.CODE)) == 0)){
  lfiles <- transform(pData(myPhenoData), CASE.CODE = paste(CASE.CODE,
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""))
  }
  lfiles <- paste(path.corpus, lfiles$CASE.CODE, sep = "/")
  lfileschecked <- sapply(lfiles, check.files)
  positions2keep <- which(!unlist(t(lapply(lfileschecked, is.null))))
  myPhenoData <- myPhenoData[positions2keep]

  Qname <- character()
  exprdata <- justRMA(filenames= paste(substr(path.corpus, 3, 60), "/",
myPhenoData$CASE.CODE[!(myPhenoData$CASE.CODE %in% Qname)],
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""),
  verbose=TRUE,
  sampleNames=myPhenoData$CASE.FILENAME[!(myPhenoData$CASE.CODE %in% Qname)],
  phenoData=myPhenoData)

##########################################################
###                                                    ###
###    Prepara Corpus                                  ###
###                                                    ###
##########################################################
### ETIQUETAS

labels2 <- phenoData(exprdata)$GlioLabels
design<- model.matrix(~factor(labels2))
colnames(design) <- c("GBM","OTHER")
rownames(design) <- exprdata$CASE.CODE

### FUNCION ALEATORIO . La variable x es el conjunto de etiquetas .
### Devuelve las etiquetas separadas en dos grupos, uno para training y otro para test.
 funcionAl <- function (x)

 {
   trainingLabSamples <- list()
   testLabSamples <- list()
   traingPosSamples <- c()
   testPosSamples <- c()
   # pasamos el vector de etiquetas de CHAR a factor
   xfactor <- factor(x)

   for(j in 1:length(levels(xfactor)))
   {
     posLab <- which(xfactor == levels(xfactor)[j])
     #  Solo nos queremos quedar con unas 10 variables más o menos

     #-------------------------------------
     #    C O N     R E P E T I C I O N   |
     #-------------------------------------
     if (length(posLab) > 10)
        permutacion <- sample(posLab,round(length(posLab)*0.5) ,replace = TRUE)
     else
        permutacion <- sample(posLab,round(length(posLab)) ,replace = TRUE)
     trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.6)]; # Separamos la
parte de Training de la de Test
     traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     testLabSamples[[j]] <- setdiff(permutacion, trainingLabSamples[[j]])

     #-------------------------------------
     #    S I N     R E P E T I C I O N   |
     #-------------------------------------
     #permutacion <- sample(posLab,round(length(posLab)*0.5))
     #trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.7)];
     #traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     #testLabSamples[[j]] <-
permutacion[(round(length(permutacion)*0.7)+1):(length(permutacion))]

     testPosSamples <- c(testPosSamples, testLabSamples[[j]])
   }
   list(traingPosSamples = traingPosSamples, testPosSamples = testPosSamples)
 }
```

```
 geneselecionados <- 10 # Cuantos genes significativos queremos que sean seleccionados
 genesPos <- list()
 diffExpressedGenesLista <- list()
 n_test <- 9
 Acierto <- vector()
 acierto_clase1 <- vector()
 acierto_clase2 <- vector()
 acierto_clase3 <- vector()
 acierto_marginal <- vector()
for (i in 1:200)
{
       print.default(i)

  rm(labels2)
  rm (design)

  labels2 <- phenoData(exprdata)$GlioLabels
  # nos guardamos las etiquetas

  design<- model.matrix(~factor(labels2))
  summary(labels2)

  colnames(design) <- c("GBM","OTHER")
  rownames(design) <- exprdata$CASE.CODE

  muestraTraingTest<- funcionAl(phenoData(exprdata)$GlioLabels)
  exprDataTraining <- exprdata[,muestraTraingTest$traingPosSamples]
  ###diseño de matriz (solo con training)
  design <- model.matrix(~factor(phenoData(exprDataTraining)$GlioLabels))
  ### lmfit (solo con training)
  eb <- eBayes(lmFit(exprs(exprDataTraining), design))
  difExpressedGenesTable <- topTable(eb, number = geneselecionados, adjust = "fdr")
  diffExpressedGenesLista[[i]] <- difExpressedGenesTable[difExpressedGenesTable[, 6] <1e-04,
1]
  genesPos[[i]] <- which(featureNames(exprDataTraining) %in% diffExpressedGenesLista[[i]])
  ### lda (solo con training)
  ### Comprobamos y limpiamos los posibles valores a 0
  for( j in 1:length(diffExpressedGenesLista))
   {
      if (length(diffExpressedGenesLista[[j]]) <= 0)
          diffExpressedGenesLista[[j]] <- "Void"
   }

  z <- lda(factor(phenoData(exprDataTraining)$GlioLabels) ~
.,data.frame(t(exprs(exprDataTraining[(diffExpressedGenesLista[[i]]), ])),row.names = NULL))

  exprDataTest <- exprdata[,muestraTraingTest$testPosSamples]
  TestResult<- predict(z,
  data.frame(t(exprs(exprDataTest[diffExpressedGenesLista[[i]],])))$class


  labels <- factor(labels2)
  TestResult <- factor(TestResult)
  levels(TestResult) <- levels(factor(labels2))
  confusMatrix <- table(labels2[1:length(TestResult)], TestResult)
  #----------------------------
  if ((!levels(TestResult)[1] %in% TestResult)&&(dim(confusMatrix)[1] < 2 )) {
        confusMatrix <- rbind(c(0,0), confusMatrix)
        acierto_clase1[[i]]   <- 0
        acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
  }
  if((!levels(TestResult)[2] %in% TestResult)||(dim(confusMatrix)[1] < 2 )){
        confusMatrix <- rbind(confusMatrix, c(0,0))
        acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
        acierto_clase2[[i]]   <- 0
  }

  if((levels(TestResult)[1] %in% TestResult)&&
     (levels(TestResult)[2] %in% TestResult)&&(dim(confusMatrix)[1] == 2 )){
      acierto_clase1[[i]]   <- (confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]))
      acierto_clase2[[i]]   <- (confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]))
     }
     n_test <- confusMatrix[[1,1]]+confusMatrix[[2,1]]+confusMatrix[[1,2]]
               +confusMatrix[[2,2]]
     Acierto[[i]]  <- (confusMatrix[[1,1]]+confusMatrix[[2,2]])/n_test
```

```
    acierto_marginal[[i]] <- (acierto_clase1[[i]]+acierto_clase2[[i]])/2
}
```

Anexo 5. Código de clasificación de grados.

```
##########################################################
###                                                    ###
###    Clasificación de grados (GBM vs.GG3 vs.GG2)     ###
###                                                    ###
##########################################################
  library(limma)
  library(MASS)
  library(annotate)
  library(geneplotter)
  library(hgu133plus2.db)
  library(GOstats)
  library(Category)
  library(e1071)
  library(affy)
  library(affyPLM)
  library(simpleaffy)
  # Funcion que comprueba que el fichero exista
  check.files <- function(x) {
  if (file.exists(x)) {
  return(x)
  }
  else return(NULL)
  }
  # Aumentamos la memoria que puede utilizar el R a 4 Mb
  memory.size(max = 4000)
  ##########################################################
  ###    Funciones necesarias                            ###
  ##########################################################

  mapping <- function(labels,classesList,mappingList,default=NA,na.keep=TRUE)
  {
        if (is.na(default))
        {              res <- labels  }
        else    { res <- rep(default,length(labels))  }

        for (i in 1:length(classesList))
        {
               res[labels %in% mappingList[[i]]] <-  classesList[[i]]
        }
        if (na.keep)
        {       res[is.na(labels)]=NA  }
        return(res)
  }

  fusiona <- function (v1, v2) {
        for (i in 1: length(v1))
              if  (v1[i] == 'other' && v2[i] == 'other'){
                           res[i] = 'other'
            }
            else {
                  if (v1[i] != 'other') {
                       res[i] = v1[i]
               }
                  else {
                       res[i] = v2[i]
               }
           }
          return (res)
  }
  ##########################################################
  ###                                                    ###
  ###    Filtramos por etiquetas GBM, GG3, GG2           ###
  ###                                                    ###
  ##########################################################
  path.corpus <- paste(".", "corpus/BASEDATOS/BDDescom", sep = "/")
  samples <- "etDB_ma_090625.csv"
  myPhenoData <- read.AnnotatedDataFrame(samples, path.corpus,sep = ";", header = TRUE)
  labels <- myPhenoData$CR.CONSENSUS.CLINICAL.DIAGNOSIS
```

```
  #Elimina los espacios en blanco de las etiquetas
  labels <- gsub(" ", "", labels)

AS2 <- 'AS2'
AS2Set <- c('DIFFUSEASTROCYTOMA9400/3','FIBRILLARYASTROCYTOMA9420/3')
OA2 <- 'OA2'
OA2Set <- c('OLIGOASTROCYTOMA9382/3')
OD2 <- 'OD2'
OD2Set <- c('OLIGODENDROGLIOMA9450/3')
Oligo <- 'OLIGO'
OligoSet <- c(OA2Set,OD2Set)
MEN <- 'MEN'
MENSet <-
c('MENINGIOMA9530/0','SECRETORYMENINGIOMA9530/0','FIBROUSMENINGIOMA9532/0','MENINGOTHELIALMENI
NGIOMA9531/0','TRANSITIONALMENINGIOMA9537/0')
GBM <- 'GBM'
GBMSet <- c('GLIOBLASTOMA9440/3','GIANTCELLGLIOBLASTOMA9441/3','GLIOSARCOMA9442/3')
MET <- 'MET'
METSet <- 'METASTASIS8000/6'
GG2 <- 'GG2'
GG2Set <- c(AS2Set,OA2Set,OD2Set)
EPE <- 'EPE'
EPESet <- c('EPENDYMOMA9391/3')
JPA <- 'JPA'
JPASet <- c('PILOCYTICASTROCYTOMA9421/1')
GG1 <- 'GG1'
GG1Set <- c(EPESet,JPASet)
AS3 <- 'AS3'
AS3Set <- c('ANAPLASTICASTROCYTOMA9401/3')
OA3 <- 'OA3'
OA3Set <- c('ANAPLASTICOLIGOASTROCYTOMA9382/3')
OD3 <- 'OD3'
OD3Set <- c('ANAPLASTICOLIGODENDROGLIOMA9451/3')
GG3 <- 'GG3'
GG3Set <- c(OA3Set,OD3Set,AS3Set)
LGG <- 'LGG'
LGGSet <- c(GG1Set,GG2Set)
HGG <- 'HGG'
HGGSet <- c(GG3Set,GBMSet)
MED <- 'MED'
MEDSet <- 'MEDULLOBLASTOMA9470/3'
LYM <- 'LYM'
LYMSet <- 'MALIGNANTLYMPHOMAS 9590/3'
### mapeo
GradosClasses <- list(GBM,GG2,GG3)
GradosList <- list(GBMSet,GG2Set,GG3Set)
GradosLabels <- mapping(labels,GradosClasses,GradosList,'other')
myPhenoData$GradosLabels <- GradosLabels

##########################################################
###                                                    ###
###    Creamos el exprdata                             ###
###                                                    ###
##########################################################
  myPhenoData <- myPhenoData[myPhenoData$GradosLabels %in% c("GBM" ,"GG3" ,"GG2")]
  if ((length(grep(".cel", myPhenoData$CASE.CODE)) == 0))
  {
  lfiles <- transform(pData(myPhenoData), CASE.CODE = paste(CASE.CODE,
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""))
  }
  lfiles <- paste(path.corpus, lfiles$CASE.CODE, sep = "/")#[myPhenoData$GradosLabels %in%
c("AS2","OA2","OD2")]
  lfileschecked <- sapply(lfiles, check.files)
  positions2keep <- which(!unlist(t(lapply(lfileschecked, is.null))))
  myPhenoData <- myPhenoData[positions2keep]

  Qname <- character()
  exprdata <- justRMA(filenames= paste(substr(path.corpus, 3, 60), "/",
myPhenoData$CASE.CODE[!(myPhenoData$CASE.CODE %in% Qname)],
  "/ma/", myPhenoData$MICROARRAY.CODE, "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,
  "/", myPhenoData$MICROARRAY.EXPERIMENT.CODE,".cel", sep = ""),
  verbose=TRUE,
  sampleNames=myPhenoData$CASE.FILENAME[!(myPhenoData$CASE.CODE %in% Qname)],
  phenoData=myPhenoData)
```

```
#########################################################
###                                                   ###
###    Prepara Corpus                                 ###
###                                                   ###
#########################################################
### ETIQUETAS

labels2 <- phenoData(exprdata)$GradosLabels
design<- model.matrix(~factor(labels2))

colnames(design) <- c("GBM" ,"GG3" ,"GG2")
rownames(design) <- exprdata$CASE.CODE

### FUNCION FIT

 funcionFit <- function (x,y)
  {
      return (eBayes(lmFit(x, y)))
  }

### FUNCION ALEATORIO . La variable x es el conjunto de etiquetas .
### Devuelve las etiquetas separadas en dos grupos, uno para training y otro para test.

 funcionAl <- function (x)
 {
   trainingLabSamples <- list()
   testLabSamples <- list()
   traingPosSamples <- c()
   testPosSamples <- c()
   # pasamos el vector de etiquetas de CHAR a factor
   xfactor <- factor(x)

   for(j in 1:length(levels(xfactor)))
   {
     posLab <- which(xfactor == levels(xfactor)[j])
     #  Solo nos queremos quedar con unas 10 variables más o menos

     #-----------------------------------
     #    C O N     R E P E T I C I O N   |
     #-----------------------------------
     #permutacion <- sample(posLab,round(length(posLab)*0.5) ,replace = TRUE)
     permutacion <- sample(posLab,length(posLab) ,replace = TRUE)
     trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.6)]; # Separamos la
parte de Training de la de Test
     traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     testLabSamples[[j]] <- setdiff(permutacion, trainingLabSamples[[j]])

     #-----------------------------------
     #    S I N     R E P E T I C I O N   |
     #-----------------------------------
     #permutacion <- sample(posLab,round(length(posLab)*0.5))
     #trainingLabSamples[[j]] <- permutacion[1:round(length(permutacion)*0.7)];
     #traingPosSamples <- c(traingPosSamples, trainingLabSamples[[j]])
     #testLabSamples[[j]] <-
permutacion[(round(length(permutacion)*0.7)+1):(length(permutacion))]

     testPosSamples <- c(testPosSamples, testLabSamples[[j]])
   }
   list(traingPosSamples = traingPosSamples, testPosSamples = testPosSamples)
   #return(traingPosSamples)
 }


 #-----------------------------------
 #    GRAN BUCLE                    |
 #-----------------------------------
 geneselecionados <- 10
 genesPos <- list()
 diffExpressedGenesLista <- list()
 n_test <- 9
 Acierto <- vector()
 acierto_clase1 <- vector()
 acierto_clase2 <- vector()
 acierto_clase3 <- vector()
```

```
 acierto_marginal <- vector()
for (i in 1:200)
{
       print.default(i)

  labels2 <- phenoData(exprdata)$GradosLabels
  # nos guardamos las etiquetas
  design<- model.matrix(~factor(labels2))
  summary(labels2)
  colnames(design) <- c("GBM" ,"GG3" ,"GG2")
  rownames(design) <- exprdata$CASE.CODE
  muestraTraingTest<- funcionAl(phenoData(exprdata)$GradosLabels)
  exprDataTraining <- exprdata[,muestraTraingTest$traingPosSamples]
  ###diseño de matriz (solo con training)
  design <- model.matrix(~factor(phenoData(exprDataTraining)$GradosLabels))
  eb <- eBayes(lmFit(exprs(exprDataTraining), design))
  difExpressedGenesTable <- topTable(eb, number = geneselecionados, adjust = "fdr") #Numero de
genes que cogemos <--
  rm(eb)
  diffExpressedGenesLista[[i]] <- difExpressedGenesTable[difExpressedGenesTable[, 8] <1e-04,
1]

  ### lda (solo con training)
  ### Comprobamos y limpiamos los posibles valores a 0
  for( j in 1:length(diffExpressedGenesLista))
   {
      if (length(diffExpressedGenesLista[[j]]) <= 0)
          diffExpressedGenesLista[[j]] <- "Void"
   }
   genesPos[[i]] <- which(featureNames(exprDataTraining) %in% diffExpressedGenesLista[[i]])
  z <- lda(factor(phenoData(exprDataTraining)$GradosLabels) ~
.,data.frame(t(exprs(exprDataTraining[(diffExpressedGenesLista[[i]]), ])),row.names = NULL))
  exprDataTest <- exprdata[,muestraTraingTest$testPosSamples]
  TestResult<- predict(z,
data.frame(t(exprs(exprDataTest[diffExpressedGenesLista[[i]],]))))$class
  labels <- factor(labels2)
  # cambiar
  TestResult <- factor(TestResult)
  levels(TestResult) <- levels(factor(labels2))
  confusMatrix <- table(labels2[1:length(TestResult)], TestResult)

  if (dim(confusMatrix)[1] < 3 ) {
         if (dim(confusMatrix)[1] < 2 ) {
             confusMatrix <- rbind(confusMatrix, c(0,0,0), c(0,0,0))
             acierto_clase1[[i]]   <-
(confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]+confusMatrix[1,3]))
             acierto_clase2[[i]]   <- 0
             acierto_clase3[[i]]   <- 0}
         if (dim(confusMatrix)[1] == 2 ) {
             confusMatrix <- rbind(confusMatrix, c(0,0,0))
             acierto_clase1[[i]]   <-
(confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]+confusMatrix[1,3]))
             acierto_clase2[[i]]   <-
(confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]+confusMatrix[2,3]))
             acierto_clase3[[i]]   <- 0 }
  }
  else{
         acierto_clase1[[i]]   <-
(confusMatrix[1,1]/(confusMatrix[1,1]+confusMatrix[1,2]+confusMatrix[1,3]))
         acierto_clase2[[i]]   <-
(confusMatrix[2,2]/(confusMatrix[2,1]+confusMatrix[2,2]+confusMatrix[2,3]))
         acierto_clase3[[i]]   <-
(confusMatrix[3,3]/(confusMatrix[3,1]+confusMatrix[3,2]+confusMatrix[3,3]))
  }
  Acierto[[i]]  <- (confusMatrix[[1,1]]+confusMatrix[[2,2]]+confusMatrix[[3,3]])/n_test
  acierto_marginal[[i]] <- (acierto_clase1[[i]]+acierto_clase2[[i]]+acierto_clase3[[i]])/3
}
```