

Análisis de datos de microarrays

Alex Sánchez-Pla y M. Carme Ruíz de Villa

Departament d'Estadística. Universitat de Barcelona.

Facultat de Biología. Avda. Diagonal 643. 08028 Barcelona. Spain.

asanchez@ub.edu; mruiz_de_villa@ub.edu

xx

xxx/xxxxx/xxx

Módulo xx

Índice

I Preliminares	6
Introducción	5
Materiales complementarios	6
Objetivos	6
1 Introducción a los microarrays	7
1.1 Antecedentes históricos	7
1.2 Los microarrays	7
1.2.1 Aplicaciones de los microarrays	8
1.3 Cómo funcionan los microarrays.....	9
1.3.1 Microarrays de dos colores	9
1.3.2 Microarrays de un color	10
1.3.3 Realización de un experimento de microarrays	11
1.3.4 Como se mide la expresión.....	13
1.4 Bioinformática de Microarrays	15
1.4.1 Software para el análisis de datos de microarrays	16
1.4.2 Bases de datos de Microarrays	20
1.5 Extensiones (1): Otros tipos de microarrays.....	20
1.5.1 Otros microarrays de ADN	21
1.5.2 Otros tipos de microarrays: Proteínas o carbohidratos ...	22
1.6 Extensiones (2): <i>NGS: Next(Now) Generation Sequencing</i>	23
1.6.1 Visión global de las tecnologías de secuenciación	24
1.6.2 De la secuenciación Sanger a la ultrasecuenciación	24
1.6.3 Tecnologías de ultrasecuenciación	25
2 El lenguaje estadístico R	28
2.1 ¿Qué son R y Bioconductor?	28
2.1.1 Descarga e instalación de R	28
2.1.2 Interfaces gráficas de usuario.....	31
2.2 Introducción al lenguaje R	31
2.2.1 Conceptos básicos del lenguaje R y su entorno	31
2.2.2 Tipos de objetos en R	33
2.2.3 Uso de R como calculadora	41
2.2.4 Entrada y salida de datos	44
2.2.5 Operaciones con vectores	45
2.2.6 Manipulación de los datos	47
2.2.7 Bucles y funciones condicionales	54

2.2.8	Gráficos en R	56
2.2.9	Operaciones con matrices de datos	63
2.2.10	Utilización de scripts	65
2.2.11	Más información	66
2.3	El proyecto Bioconductor	66
2.3.1	Objetivos de Bioconductor	66
2.3.2	Paquetes	67
2.3.3	Programación orientada a objetos en R y Bioconductor ..	67
2.3.4	Dos clases importantes: <code>expressionSet</code> y <code>affyBatch</code>	69
2.3.5	Primeros pasos: Instalación, cursos, <i>vignettes</i>	70
2.3.6	Listado de paquetes usados o citados en el texto:	71
3	Fundamentos de Estadística	77
3.1	Introducción	77
3.2	Análisis descriptivo	78
3.2.1	Variables categóricas	78
3.2.2	Variables numéricas:	79
3.2.3	Gráficos:	80
3.2.4	Estadísticos descriptivos	82
3.3	Distribuciones de probabilidad importantes en estadística	83
3.3.1	Distribuciones discretas	83
3.3.2	Distribuciones continuas	84
3.4	Inferencia estadística	90
3.4.1	Contrastes de hipótesis	90
3.4.2	Test t de una muestra	93
3.4.3	Test t de dos muestras con varianzas distintas	94
3.4.4	Test t de dos muestras con varianzas iguales	95
3.4.5	Test F de igualdad de varianzas	95
3.4.6	Test Binomial	96
3.4.7	Test Chi-cuadrado	97
3.4.8	Test de Normalidad	99
3.4.9	Test de rangos de Wilcoxon	99
3.5	Corrección para pruebas múltiples (<i>Multiple testing</i>)	100
3.6	Análisis de la varianza	101
3.6.1	Análisis de la varianza de un factor	101
3.6.2	ANOVA de más de un factor	104
3.7	Introducción a los métodos multivariantes	105
3.7.1	Análisis de Componentes Principales	105
3.7.2	Análisis de conglomerados (Cluster Analysis)	106
II	Análisis de datos de microarrays	109
4	El proceso de análisis de datos de microarray (MDA)	110
4.1	Introducción	110
4.2	Tipos de estudios	110
4.2.1	Comparación de grupos o <i>Class comparison</i>	110

4.2.2	Predicción de clase o <i>Class prediction</i>	111
4.2.3	Descubrimiento de clases o <i>Class discovery</i>	111
4.2.4	Otros tipos de estudios	112
4.3	Algunos ejemplos concretos	112
4.3.1	Estudio de procesos regulados por citoquinas	112
4.3.2	Clasificación molecular de la leucemia.....	113
4.3.3	Efecto del estrogeno y el tiempo de administración.....	113
4.3.4	Efecto del CCL4 en la expresión génica	113
4.3.5	Análisis de patrones en el ciclo celular	114
4.3.6	Recapitulación	114
4.4	El proceso de análisis de microarrays	114
5	Diseño de experimentos de microarrays	117
5.1	Fuentes de variabilidad.....	117
5.2	Principales conceptos en Diseño de Experimentos	118
5.3	Principios básicos en el diseño del experimento	118
5.4	Replicación	119
5.4.1	Potencia y tamaño de la muestra	119
5.4.2	Pooling	120
5.5	Diseños experimentales para microarrays de dos colores	121
6	Exploración de los datos, control de calidad y preprocesado ..	124
6.1	Introducción	124
6.1.1	Nivel de análisis y tipo de microarray	124
6.1.2	Datos de partida.....	125
6.2	Gráficos para la exploración y control de calidad	128
6.2.1	Control de calidad con gráficos estadísticos generales....	128
6.2.2	Gráficos de diagnóstico para microarrays de dos colores ..	131
6.2.3	Gráficos de diagnóstico para microarrays de un color.....	133
6.3	Normalización de arrays de dos colores	136
6.3.1	Normalización global	137
6.3.2	Normalización dependiente de la intensidad	137
6.4	Resumen y normalización de microarrays de Affymetrix	137
6.4.1	Métodos originales de Affymetrix	139
6.4.2	El método RMA (Robust Multi-Array Average)	140
6.5	Filtraje no específico	141
7	Selección de genes diferencialmente expresados	142
7.1	Introducción	142
7.1.1	Medidas <i>naturales</i> para comparar dos muestras	143
7.1.2	Selección de genes diferencialmente expresados	145
7.1.3	Potencia y tamaño muestral	149
7.1.4	El problema de la multiplicidad de tests (“multiple testing”	149
7.2	Modelos lineales para la selección de genes: <code>limma</code>	151
7.2.1	El modelo lineal general	152
7.2.2	Ejemplos de situaciones <i>modelizables</i> linealmente	152
7.2.3	Ejemplo 2: Comparación de tres grupos	153

7.2.4	Estimación e inferencia con el modelo lineal.....	161
7.2.5	Modelos lineales para Microarrays	162
7.2.6	Implementación y ejemplos	163
8	Después de la selección: Análisis de listas de genes	166
8.1	Introducción	166
8.2	Análisis comparativo de listas de genes	166
8.2.1	Ejemplo: comparación de listas en el caso <code>celltypes</code>	167
8.3	Anotación de los resultados	168
8.4	Visualización de los perfiles de expresión	172
8.5	Análisis de significación biológica.....	174
9	Caso resuelto: Selección de genes diferencialmente expresados	178
9.1	Introducción	178
9.1.1	El ejemplo	179
9.1.2	Directorios y opciones de trabajo	179
9.2	Obtención y lectura de los datos	180
9.2.1	Los datos	180
9.2.2	Lectura de los datos	180
9.3	Exploración, Control de Calidad y Normalización	181
9.3.1	Exploración y visualización	181
9.3.2	Control de calidad	183
9.3.3	Normalizacion y Filtraje	184
9.4	Selección de genes diferencialmente expresados	187
9.4.1	Análisis basado en modelos lineales	188
9.4.2	Comparaciones múltiples	192
9.4.3	Anotación de resultados	193
9.4.4	Visualización de los perfiles de expresión	194
10.	Métodos avanzados: Busca de patrones y predicción	196
10.1	Descubrimiento de grupos en datos de microarrays	196
10.1.1	Principios y métodos para el descubrimiento de clases ...	197
10.1.2	Consistencia de la agrupación	201
10.2	Diagnósticos moleculares y métodos de clasificación	203
10.2.1	La selección de variables para la clasificación.....	205
11.	Caso resuelto: Descubrimiento de clases	208
11.1	Clustering y visualización	208
11.1.1	Pre-procesamiento de los datos	208
11.1.2	Agrupación jerárquica de las muestras	210
11.1.3	Agrupación de genes por el método de las <code>k-means</code>	212
11.1.4	Anotación de resultados	214
Resumen	216

Part I

Preliminares

Introducción

Este documento presenta una introducción al análisis de datos de alto rendimiento, particularmente los microarrays de expresión, tal como se está llevando a cabo por la comunidad bioinformática a principios de la segunda década del siglo XXI.

El análisis de microarrays ha revolucionado la biología de principios de este siglo en cuanto que ha conllevado un cambio de paradigma: se ha pasado de poder estudiar los genes de uno en uno a poder hacerlo “todos a la vez”. Esto, sin embargo no ha sido sin coste, dado que los datos que se han ido generando presentan tanto problemas técnicos como una considerable cantidad de ruido. La eliminación de estas fuentes de error puede llevarse a cabo usando métodos estadísticos, que también se han revelado de gran importancia a la hora de analizar los datos, ya sea para seleccionar genes relevantes, para buscar perfiles de variación común entre los genes o para construir clasificadores a nivel molecular.

Por todo lo anterior el análisis de datos de microarrays es una disciplina que se encuentra a caballo entre la estadística y la bioinformática y su dominio requiere la comprensión de los procesos biotecnológicos que generan los datos, el manejo de las herramientas bioinformáticas necesarias para manipular la información y la comprensión de los métodos estadísticos subyacentes a los análisis.

Con estos requisitos un manual para el análisis de microarrays tiene que ser necesariamente extenso en tanto que tendrá que tratar mínimamente los tres aspectos considerados.

El texto que se presenta a continuación es el fruto de la experiencia en el aula –presencial y virtual– de más de 5 años de docencia en este tema y pretende buscar un equilibrio entre la formación y la información para lo cual se han seleccionado cuidadosamente los temas y puntos a tratar de forma que el alumno pueda adquirir un sólido conocimiento sin que los detalles le hagan perder la perspectiva en la que, probablemente, sea su primera incursión en este tema.

Aunque no parezca lo usual en este tipo de textos en este caso se hace necesaria una pequeña sección de agradecimientos. La versión final de este documento no habría llegado a buen puerto sin la desinteresada colaboración de muchas personas. Ferran Briansó, Xavier de Pedro, Eudald Illa, Ricardo Gonzalo y Josep Gregori han contribuido con sus comentarios a mejorar muchas partes del texto y con su ayuda a detectar un sinnúmero de pequeños problemas. Marta Casals, Oriol Rua y Silvia Cardona han invertido un buen número de horas en revisar y corregir imágenes texto y código. Sin la ayuda de todos ellos este documento no habría visto la luz. Los errores no detectados corresponden naturalmente a los

autores.

Materiales complementarios

Un texto de estas características está plagado de programas que –esperamos– el lector puede querer probar por si mismo para lo que puede además necesitar los datos de los ejemplos que desea reproducir. Con el fin de facilitar esta forma de aprendizaje todos los códigos del texto así como los datos utilizados y otros materiales de interés se han colocado en una página web que se mantendrá en paralelo con el libro.

Su dirección es:

<http://ueb.vhir.org/cursos/Microarrays>

Objetivos

1. Objetivo 1: Comprender la tecnología de los microarrays y conocer sus diversas variantes.
2. Objetivo 2: Adquirir conocimientos básicos de R y estadística que permitan llevar a cabo de forma comprensiva análisis básicos de microarrays.
3. Objetivo 3: Conocer los principales problemas que pueden resolverse con microarrays
4. Objetivo 4: Saber llevar a cabo análisis básicos con datos reales.
5. Objetivo 5: Conocer las limitaciones de los microarrays y su relación con las nuevas tecnologías de alto rendimiento como la ultrasecenciación.

1. Introducción a los microarrays

1.1 Antecedentes históricos

La biología molecular ha estado interesada desde sus comienzos en poder determinar el nivel de expresión de los genes integrados en el genoma humano. Para ello dispone desde hace años de múltiples técnicas para medir estos niveles, tales como el **Northern blot** (a nivel de ARN) o el **Western blot** (a nivel de proteína).

Especial interés ha tenido siempre sobre las otras biomoléculas el estudio de los niveles del ARN transcrit. Hace unos años se acuñó el término de *transcriptómica* para referirse al estudio del ARN en cada una de sus formas. La transcriptómica ha contribuído a tener una mejor comprensión de la patogénesis de las enfermedades. En el año 2009 Wang y colaboradores ([51]) definieron el transcriptoma como el conjunto completo de tránscritos en una célula y su cantidad en un momento determinado del desarrollo o en una determinada condición fisiológica. Como objetivos de la transcriptómica se han definido tres principales:

- 1) Catalogar todas las especies de tránscritos, incluyendo ARNm, ARN no codificante y pequeños ARNs.
- 2) Determinar la estructura transcripcional de los genes, en términos de sus puntos 5' de inicio y 3' de finalización, las modificaciones post-transcripcionales y los patrones de splicing.
- 3) Cuantificar los diferentes niveles de expresión de cada tránsrito durante el desarrollo y bajo diferentes condiciones.

El transcriptoma

el conjunto completo de tránscritos en una célula y su cantidad en un momento determinado del desarrollo o en una determinada condición fisiológica

El interés de la transcriptómica no solamente se ha centrado en el desarrollo de nuevas tecnologías que mejoran su estudio, sino también en el desarrollo de nuevos métodos de extraer la gran cantidad de información que se genera con estas nuevas técnicas.

1.2 Los microarrays

Una de las técnicas que revolucionó el estudio del transcriptoma fueron los microarrays. Un microarray es un artefacto para la realización de experimentos que permite estudiar simultáneamente múltiples unidades, b que representan los genes, proteínas o metabolitos, sobre un sustrato sólido de cristal, plástico o sílice,

y expuestos a la acción de las moléculas diana cuya expresión se desea analizar (ver Figura 1).

Figura 1. Microarrays

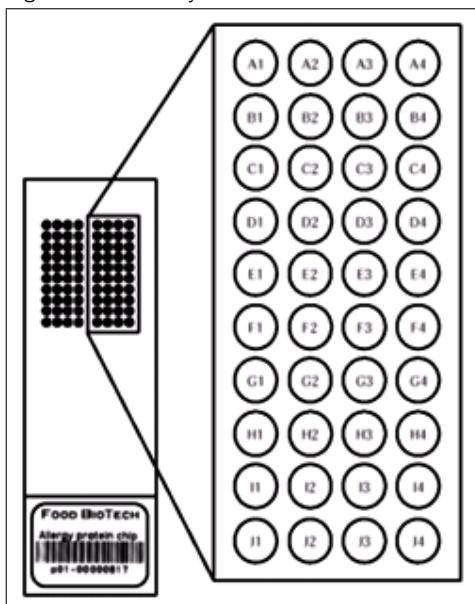


Figura 1

Imagen de un microarray

La utilización de los microarrays en la última década ha generado inmensas cantidades de datos útiles para el estudio y el desarrollo de enfermedades, dando a conocer múltiples mapas de expresión génica, encontrar biomarcadores o construir firmas génicas para determinadas enfermedades.

Lo que caracteriza a los nuevos métodos utilizados para estudiar el transcriptoma no es lo que pueden medir, sino la cantidad de mediciones simultáneas que pueden realizar. Mientras que hasta hace apenas una década se estudiaban los genes uno a uno en profundidad, a partir del uso de estas nuevas tecnologías se pueden estudiar muchísimos genes a la vez, pero en contrapartida con mucho menos detalle y más ruido.

Los microarrays, hoy una metodología bien consolidada, han sido cruciales para concebir una nueva manera de estudiar el transcriptoma, especialmente en el campo de la expresión génica. Después de ellos han venido otras técnicas que tienen en común con ellos el “alto rendimiento” es decir la capacidad para medir muchas variables –cientos o miles– a la vez. Entre estas técnicas podemos destacar los arrays de SNPs, los microRNAs, la metilación y especialmente la secuenciación de nueva generación.

1.2.1 Aplicaciones de los microarrays

La tecnología de los microarrays se ha aplicado a una inmensa variedad de problemas, desde el estudio de enfermedades como el cáncer o la esclerosis múltiple al de los ritmos circadianos de las frutas. Entre otros temas los microarrays se han aplicado a:

- Estudio de genes que se expresan diferencialmente entre varias condiciones (sanos vs enfermos, mutantes vs salvajes, tratados vs no tratados) ([6, 26, 33]).
- Clasificación molecular en enfermedades complejas ([24, 3]).
- Identificación de genes característicos de una patología (firma o “signature”) ([17]).
- Predicción de respuesta a un tratamiento ([37]).
- y una gran variedad de otros temas

1.3 Cómo funcionan los microarrays

En términos generales los microarrays funcionan mediante la hibridación de una sonda específica (“probe”) y una molécula diana (“target”). La hibridación que ha tenido lugar se detecta mediante fluorescencia y se visualiza con la ayuda de un escáner. Los niveles de fluorescencia detectados reflejan la cantidad de moléculas diana presentes en la muestra problema.

Existen diferentes tipos de microarrays según la naturaleza del “target” que se está estudiando. Podemos encontrar microarrays de proteínas, de tejidos, de ADN o de ARN (también llamados de expresión). En este curso nos centraremos en los microarrays de expresión génica aunque al final del capítulo se hace mención de algunos de los otros tipos de microarrays que existen.

Una clasificación habitual de los microarrays es por el número de muestras que se hibridan simultáneamente. Distinguimos:

- 1) Microarrays de dos colores o “spotted arrays”
- 2) Microarrays de un color o arrays de oligonucleótidos.

1.3.1 Microarrays de dos colores

Estos arrays aparecieron a mediados de los años 90 ([44]) y están basados en la *hibridación competitiva* de dos muestras, cada una de las cuales ha sido marcada con un marcador fluorescente diferente (normalmente Cy3 y Cy5, verde y rojo respectivamente).

En el array se imprimen las sondas (“una sonda≈ un gen”) cuyas secuencias se obtienen de información almacenada en bases de datos de secuencias como GenBank, dbEST, etc.

Figura 2. Esquema del funcionamiento de un microarray de dos colores

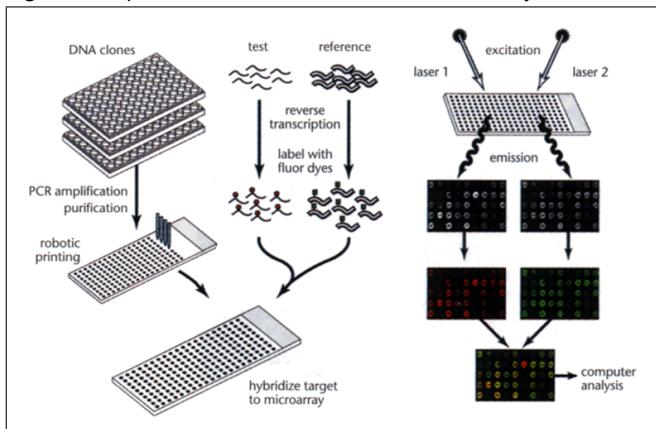


Figura 2

En los micrarray de dos colores las sondas son sintetizadas *in vitro* y depositadas directamente sobre una superficie de cristal.

El ARN de las muestras problema es extraído y posteriormente marcado con los marcadores fluorescentes, según a que grupo a comparar pertenezcan. Las muestras marcadas se mezclan y se hibridan sobre el array. La hibridación consiste en combinar la muestra (target) y el microarray (sondas) y dejarlos un tiempo en una cámara de hibridación a una temperatura y agitación determinadas. Las sondas que tengan secuencias complementarias en las muestras, se hibridarán con ellas y quedarán fuertemente adheridas. Pasadas unas horas se lava el microarray para eliminar los “targets” que no se hayan hibridado. Después de la hibridación el array se ilumina con un láser que provoca que el marcador fluorescente emita fluorescencia de uno u otro color generando dos imágenes que se superpondrán para su análisis conjunto. La cantidad de fluorescencia generada es proporcional a la cantidad de ARNm presente en la muestra problema. El resultado final es un valor que representa el nivel de expresión de una muestra respecto a la otra por lo que se le denomina expresión *relativa*.

1.3.2 Microarrays de un color

Como su propio nombre indica, en estos arrays las muestras están marcadas únicamente con un marcador fluorescente. En cada array solamente se hibrida una muestra, por lo que no se da la hibridación competitiva como pasaba en los arrays de dos colores. El valor que se obtiene después de iluminar el array con el láser es una medida numérica que se obtiene directamente del escáner, es decir no está referida al valor de otra muestra por lo que recibe el nombre de expresión *absoluta*.

Fabricación de los arrays de oligonucleótidos

La empresa Affymetrix (Santa Clara, California) es la casa comercial líder en la manufacturación y venta de este tipo de microarrays. Affymetrix sintetiza las sondas directamente sobre el chip mediante un proceso llamado fotolitografía. Este proceso consiste en la adición cíclica de los cuatro nucleótidos (adenina, timina, citosina y guanina) sobre la superficie rígida, donde existen ancladas

unas especies químicas reactivas que se protegen y desprotegen para añadir el nucleótido deseado, mediante ciclos de luz y oscuridad. Así se consigue la síntesis de oligonucleótidos de unos 25-mer (“mer”=bases) de longitud. En la figura 3 se esquematiza el proceso.

Figura 3. Microarrays de oligonucleótidos

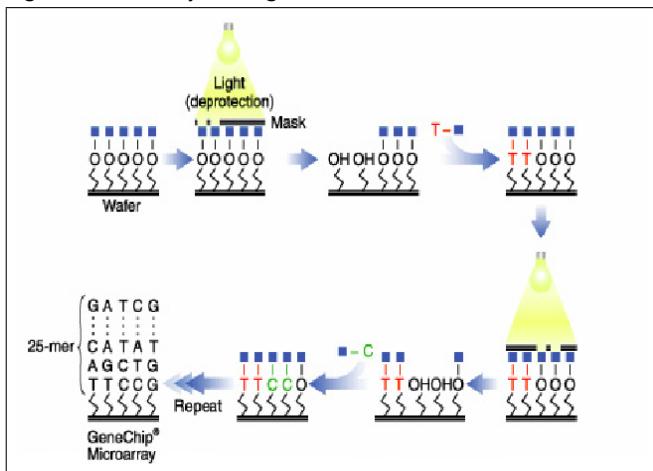


Figura 3

Esquema del proceso de fabricación de arrays de oligonucleótidos.

Cada oligonucleótido sintetizado de 25-mer (es decir 25 bases) se denomina sonda (“probe”). Alrededor de 40x107 de estas sondas se agrupan en una celda llamada “probe cell”. Las “probe cell” están organizadas en parejas, “probe pairs”, donde se combinan un “Perfect Match” (PM) -cuya secuencia de 25-mer coincide perfectamente con una parte del gen- y un “Mismatch” (MM) -idéntico al PM, excepto en el nucléotido central que no coincide (ver la figura 4)

De 11 a 20 “probe pairs” se agrupan para formar un “probe set”. Diferentes “probe sets” se distribuyen a lo largo del array, y son las que definen a qué gen se unen y las que definen los niveles de expresión detectados.

El Mismatch (MM) se creó originalmente para que proporcionara una medida de la unión inespecífica (es decir lo que se hibridara con esta secuencia “parecida” pero distinta a la original se consideraría ruido). Aunque algunos algoritmos originales de Affymetrix todavía lo utilizan, hoy en día ha caído en desuso, muchos algoritmos modernos como RMA –que se explicará más adelante– ya no lo utilizan y en las nuevas versiones de los arrays (modelos “Hugene” por ejemplo) ya no se incluyó.

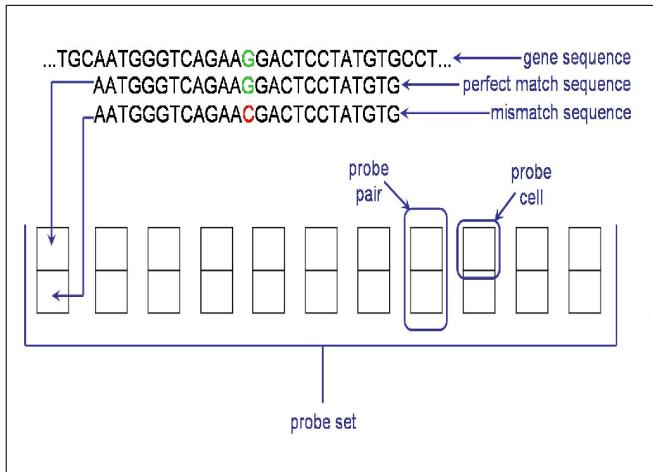
1.3.3 Realización de un experimento de microarrays

Tanto en los arrays de un color como en los de dos colores el material de partida es el ARN. Es muy importante controlar la calidad y la cantidad del ARN, ya que puede influir directamente en la calidad final de los resultados. Para ello se utiliza un equipo llamado “Bioanalyzer” (Agilent Technologies, Santa Clara, California) que proporciona entre otros parámetros un valor llamado “RNA Integrity Number (“RIN”)” que sirve para decidir si la calidad del ARN es suficientemente buena

Figura 4

Estructura de un grupo de sondas (“probe sets”) de un array tipo “genechip” de Affymetrix en el que se muestra los distintos términos definidos: “probe-set”, “probe-pair”, “Perfect-Match”, “Mismatch”.

Figura 4 Estructura de un grupo de sondas (“probe sets”).



como para que valga la pena usarlo para hibridar un microarray. Este valor varía entre 0 y 10 y en general suele exigirse un valor mínimo de por ejemplo 7 u 8.

Una vez decidido que la calidad del ARN de la muestra es aceptable la cantidad inicial de ARN es amplificada unas 25 veces utilizando enzimas y cebadores específicos.

Posteriormente el ARN es marcado con una molécula fluorescente (ficoeritrina en este caso) y fragmentado en trozos más pequeños que se puedan unir a las “probes” fijadas en el array, mediante el proceso de hibridación. Al igual que en los arrays de dos colores, el array hibridado es iluminado en un escáner mediante un láser, para que la ficoeritrina emita luz fluorescente. La fluorescencia registrada de cada “probe set”, es directamente proporcional a la cantidad de ARN presente en la muestra inicial de cada gen. La figura ?? es una representación esquemática del uso de microarrays de un color.

Figura 5 Esquema del funcionamiento de un microarray de un color.

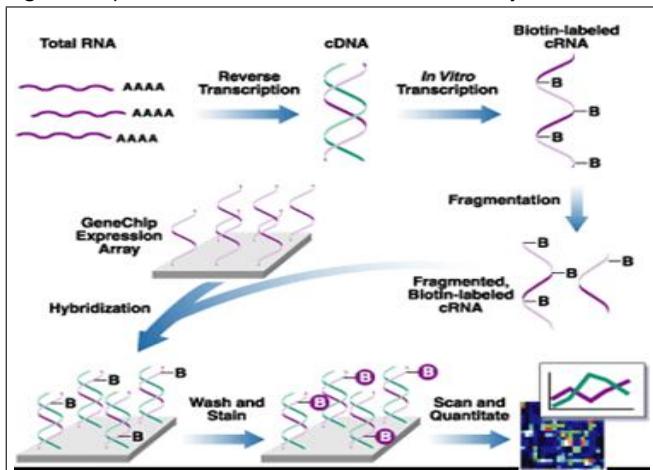


Figura 5

Esquema del funcionamiento de un microarray de Affymetrix o de un sólo color.

1.3.4 Como se mide la expresión

Los microarrays permiten cuantificar la expresión de los genes a través de la intensidad de la fluorescencia que es capturada por los escáneres. Las imágenes se convierten en valores por un proceso que no es objeto de discusión en este curso, pero que puede ser considerado relativamente fiable y estable (véase, por ejemplo [44]).

Cada tecnología genera diferentes tipos de imágenes y éstas generan diferentes valores que deben ser tratados adecuadamente para proporcionar alguna clase de estimaciones de una misma variable: la *expresión génica*.

Tal como se ha indicado anteriormente una de las principales diferencias entre arrays de uno y dos colores es que estos últimos se basan en la *hibridación competitiva* de las dos muestras mientras que los de un color sólo miden cuanta muestra se hibrida con las sondas del chip. En consecuencia en los arrays de dos colores se mide *cuánto se expresa un gen en una muestra respecto a la otra* lo que tiene un sentido biológico en términos de “sobre–expresión” (por ejemplo, si un gen se expresa dos veces más en una condición que en la otra puede deducirse que está activado o sobre–expresado) o “sub–regulación” (si el gen se expresa la mitad en una condición que en otra puede decirse que está inhibido o regulado negativamente). En los arrays de un color en cambio tan solo se mide *cuánto se expresa un gen* en una escala que carece de sentido biológico.

Medición de la expresión relativa en arrays de dos colores

Cuando la imagen obtenida en un *microarray de dos colores* es analizada (“cuantitativamente”), en cada spot se generan algunos valores (ver figura 6). Aunque esto depende del software utilizado, básicamente consiste en (i) Medidas de señales, Rojo (R) o Verde (G) para cada canal , (ii) Medidas de background, R_b , G_b , que intentan proporcionar una medida de la fluorescencia no debida a hibridación, y (iii) algunas medidas de calidad para el spot.

Estas cantidades pueden utilizarse para proporcionar medidas sencillas del ratio de expresión:

$$M = \frac{R}{G}, \quad (1)$$

o el *background* corregido del ratio de expresión:

$$M = \frac{R - R_b}{G - G_b}. \quad (2)$$

Es habitual la utilización del logaritmo en base 2 de esta cantidad como el resultado final de *expresión relativa*. Esto se debe principalmente a dos motivos: por un lado, los datos de expresión se aproximan mejor con una distribución log-normal, y por otro lado, la utilización de logaritmos simetriza las diferencias haciendo más fácil la interpretación.

Figura 6.

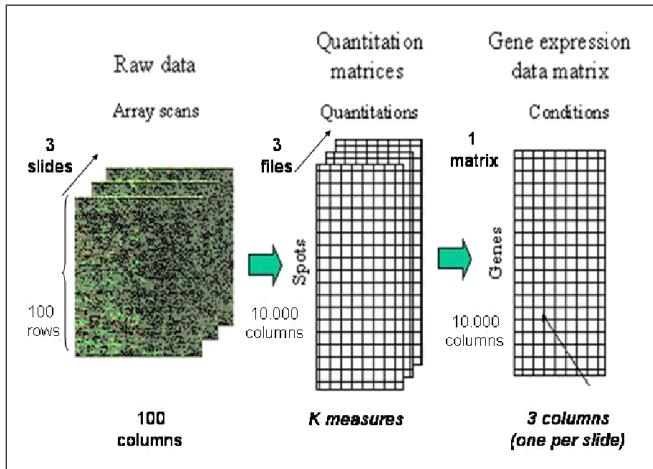


Figura 6

Esquema del tratamiento de las imágenes para su cuantificación.

La tabla 1 muestra de forma simplificada el aspecto que puede tener una matriz de expresión relativa en la que para estudiar 6 muestras apareadas, tres de tejidos sanos y tres tumorales, se ha hibridado cada tumor contra su control “normal” dando lugar a una matriz de expresión de 5 genes y 3 columnas de expresiones relativas.

Tabla 1. Ejemplo simplificado que muestra como podría ser una matriz de expresiones relativas obtenidas de 6 muestras apareadas (3 individuos) y 5 genes

	log (Tum1/Norm1)	log (Tum2 /Norm2)	log (Tum3/Norm3)
Gene 1	0.46	0.80	1.51
Gene 2	-0.90	0.06	-3.20
Gene 3	0.15	0.04	0.09
Gene 4	0.60	1.06	1.35
Gene 5	-0.45	-1.03	-0.79

Medición de la expresión absoluta en arrays de un color

Los arrays de Affymetrix representan cada gen como un conjunto de sondas, cada una de las cuales se corresponde con una fragmento corto de un gen. De hecho, tal como se ha dicho, no se trata de sondas sino de parejas de sondas formadas por un “Perfect Match” que corresponde a la cadena de ADN original y un “Mismatch” en las que ha cambiado su nucleotido central.

La idea subyacente en esta aproximación es que cualquiera que hibrida con la sonda de mismatch no debería representar una expresión real sino que representa lo que se denomina “background” o señal de fondo.

En los inicios de ésta tecnología la compañía Affymetrix sugirió combinar ambas medidas en lo que puede verse como una medida de expresión corregida para la señal de fondo. La fórmula utilizada ha evolucionado, pero, una estimación sencilla de las primeras versiones es:

$$Avg.diff = \frac{1}{|A|} \sum_{j \in A} (PM_j - MM_j), \quad (3)$$

donde A es el conjunto de pares de sondas cuyas intensidades no se desvian más de tres veces de la desviación estandar de la principal intensidad entre todas las sondas.

La tabla 2 muestra de forma simplificada el aspecto que puede tener una matriz de expresión relativa en la que para estudiar 6 muestras apareadas, tres de tejidos sanos y tres tumorales, se ha hibridoado cada tumor y cada control “normal” en un array separado dando lugar a una matriz de expresión de 5 filas (una por gen) y 6 columnas de expresiones absolutas.

Tabla 2. Ejemplo simplificado que muestra como podría ser una matriz de expresiones absolutas obtenidas de 6 muestras apareadas y 5 genes.

	log (Tum 1)	log (Tum 2)	log (Tum 3)	log (Norm 1)	log (Norm 2)	log (Norm 3)
Gene 1	5.10	6.9	6.6	6.4	7.8	4.3
Gene 2	6.97	8.74	7.89	8.03	9.70	5.63
Gene 3	4.44	6.89	6.41	6.02	7.47	4.08
Gene 4	6.43	8.13	8.56	7.14	7.63	4.81
Gene 5	8.18	10.44	13.29	7.85	9.60	5.29

La mayor diferencia entre estas dos maneras de medir la expresión no está en la fórmula específica, que ha evolucionado en ambos casos, sino en el hecho que, mientras que en los chips de Affymetrix se tiene un único valor de expresión para cada condición, en los arrays de dos colores se trabaja con una medida de la expresión relativa entre dos condiciones. Aunque Affymetrix permite estimaciones más precisas, las expresiones relativas tienen una mejor interpretación intuitiva.

1.4 Bioinformática de Microarrays

El crecimiento en el uso de la experiencia en microarrays en la última década ha sido en paralelo por los desarrollos necesarios en la metodología –nuevos métodos para modelar y analizar los datos se requieren a menudo– y la bioinformática –nuevas herramientas necesarias para implementar los métodos, así como el almacenamiento, el acceso o la organización de la mayor parte creciente de datos disponibles. Esto nos lleva a considerar dos aspectos muy importantes relacionados con el análisis de microarrays de datos:

- 1) ¿Qué software está disponible para analizar los datos de microarrays?

2) ¿Qué sistemas de base de datos están disponibles para almacenar y manipular los datos de microarrays tanto a nivel local como global?

Este tema puede ser considerado complementario, pero necesario para poner en práctica los puntos discutidos en el documento de manera que una breve presentación de los sistemas de software y base de datos se presentan a continuación.

1.4.1 Software para el análisis de datos de microarrays

Supongamos que un estadista quiere involucrarse en el análisis de datos de microarrays y después de leer un poco este entiende lo que hay que hacer. Una pregunta obvia es “¿Qué herramienta debo usar?” Como la mayoría de los profesionales en el campo que está familiarizado con varios paquetes y probablemente tiene algunas preferencias.

Después de algunas búsquedas en Google, es obvio que haya varias posibilidades

- Para utilizar los paquetes estadísticos estándar –SPSS o SAS– y analizar los datos que deben haber sido procesados o exportados del texto.
- Para usar una de las muchas herramientas disponibles gratuitamente, ya sea web o de base local.
- Para contar con extensiones específicas de microarrays para el análisis de datos tales como el Proyecto Bioconductor.
- Para comprar uno de los programas comerciales existentes.

Como es habitual cada opción tiene aspectos positivos y negativos. El uso de paquetes estadísticos estándar –SPSS o SAS– tiene la curva de aprendizaje más corta, pero no permite hacer la mayoría de los pre-pasos de procesamiento, tales como la normalización o resumen, por lo que debe combinarse con otro software. Además, si uno desea hacer un ANOVA o K-means están bien, pero si lo que uno quiere hacer es aplicar métodos específicos, tales como SAM o ajustes locales–FDR serán insuficientes. Algunos paquetes de estadística como S+ o SAS han desarrollado extensiones de gran alcance para el análisis de datos de microarrays.

Software libre o de código abierto

Existen numerosas herramientas gratuitas para el análisis de Microarrays. Algunas de ellas, además, llevan la posibilidad (libertad) de modificar el programa porque incluyen el código fuente de programa (por tanto, se las llamas de “software de

código abierto” o de “software libre”, ver: <http://www.gnu.org/philosophy/free-sw.html>). Y otras, en cambio, ponen restricciones a como permiten usar el programa, restringen la libertad de modificar el código fuente del programa, y por tanto, aunque sean gratuitas, se las llama de software propietario, no-libre o cerrado, por contraposición al software anterior.

Cualquiera de estas herramientas gratuitas puede conllevar un esfuerzo considerable en aprender a utilizarlas, pues tienen diferentes grados de madurez, y de facilidad para que nuevos usuarios las utilicen. Asimismo, puede pasar que estas herramientas no sigan estándares comunes de organización de los datos o flujos de trabajo, lo que el aprendizaje de uno no suele ayudar al aprendizaje de otro. Y cabe la posibilidad de que, como herramientas gratuitas o demasiado jóvenes, presenten una mayor tasa de errores que lo deseado. En cualquier caso, a menudo pueden resultar útiles para un análisis exploratorio de nuestros datos de microarrays, o para el mundo de la enseñanza, en especial si hay una comunidad de usuarios lo suficientemente grande de esa herramienta para garantizar que no hay demasiados problemas básicos con su uso. Pero si se desea usarlos repetidamente para la realización de estudios de media a alta complejidad, pueden resultar ser insuficiente, ya sea porque carecen de métodos, porque son ineficientes o simplemente porque no tienen la capacidad de programación para automatizar tareas repetitivas.

Algunos listados de estas herramientas son:

- http://mybio.wikia.com/wiki/Microarray_software
- http://mybio.wikia.com/wiki/Microarray_software_tools
- <http://seqanswers.com/wiki/Software/list>

A pesar de estas críticas, los programas libres especialmente (más allá de los únicamente “gratuitos”), pueden ser una forma suave para introducirse a los análisis de datos de microarrays. Para guiar a un usuario inexperto comentamos brevemente algunas de nuestras herramientas libres favoritas.

- *BRB array tools* es un complemento de Excel-ins que combina **R**, **C** y Java para hacer los cálculos y utiliza Excel para interactuar con el usuario – lo que significa que sólo está disponible para usuarios de Windows. Es proporcionada por The Biometrics Research Branch del Instituto Nacional del Cáncer (EE.UU.). Éste se complementa con tutoriales y una base de datos y de estudios para estudios reales preparados para ser utilizados con estos. Suele ser muy atractivo a primera vista, especialmente cuando se utiliza con sus propios ejemplos. Sin embargo la creación de un nuevo análisis desde el principio no es una tarea fácil y lo peor es que tiende a chocar en una dura forma con mensajes de Visual Basic cripticos, especialmente si se utilizan en los ordenadores con las versiones no inglesas de Windows.

- *TM4* es un conjunto de cuatro programas de libre escritos en Java y se ejecutados en sistemas Linux y Windows desarrollados por el instituto TIGR (ahora J. Craig Venter). Aunque un poco viejo y relativamente sesgado hacia los arrays de dos colores, para el cual fue desarrollado originalmente, es muy robusto (se bloquea mucho menos que BRB) y ofrece capacidades de análisis, no sólo (**MeV**), sino también el análisis de imágenes (**Spotfinder**), la normalización separado (**MIDAS**) y un sistema de base de datos (**MADAM**) para almacenar los experimentos.
- Un grave inconveniente de las herramientas anteriores es su sesgo histórico hacia los microarrays de dos colores lo que implica que se pierden (a partir de comienzos de 2008) los métodos de preprocesamiento importante como **RMA**. Una buena –fácil de usar– alternativa para los primeros pasos de control de calidad y pre–procesamiento de los chips de Affymetrix es ofrecido por la misma compañía. Se llama *Consola de expresión* y se puede descargar desde la web de Affymetrix después de la inscripción gratuita.
- *The http://babelomics.bioinfo.cipf.es/* es un conjunto integrado de herramientas para el análisis de datos de microarrays disponible en la web. Babelomics ha sido diseñado para proporcionar una intuitiva interfaz basada en Web que ofrece diversas opciones de análisis de la etapa inicial de pre–procesamiento (normalización de Affymetrix y experimentos de microarrays de dos colores y otras opciones de pre–procesamiento), hasta el paso final de la elaboración de perfiles funcionales de la experiencia (con gene Ontology, pathways, PubMed resúmenes, etc), que incluyen diferentes posibilidades de agrupación, de predicción de genes de selección de clase y serie-comparativo de gestión de la hibridación genómica.

Figura 7. Software de análisis de la expresión génica o Babelomics

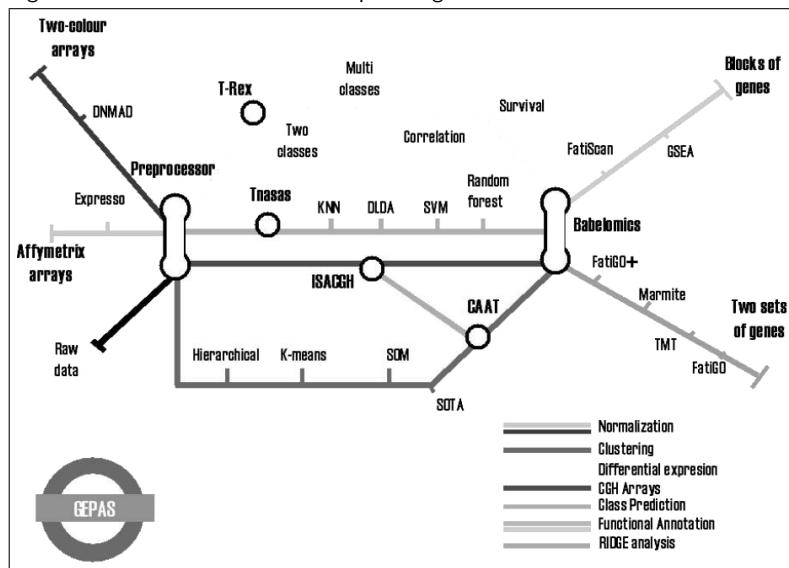


Figura 7

Un mapa de las funcionalidades de “Babelomics” organizados como en una línea de metro. Un usuario normalmente debe comenzar en alguna parte de la izquierda del mapa y finalizar en algún lugar de la derecha.

El proyecto bioconductor

Una de las opciones para el análisis de los datos mencionados anteriormente es la combinación de un software estándar, tales como **Matlab**, **Mathematica** o **R** con librerías específicas diseñadas para el análisis de microarrays. Aunque existen algunas extensiones de Matlab para el análisis de microarrays es con **R** que esta complementariedad ha alcanzado dimensiones inesperadas. El proyecto Bioconductor (<http://www.Bioconductor.org>) comenzó en 2001 como un proyecto de código abierto y libre desarrollo de software para el análisis y comprensión de datos genómicos. Su gran éxito ha hecho crecer a partir de poco más de una docena de paquetes a cientos de ellos. Casi todas las técnicas disponibles en el análisis de microarrays tienen su propio paquete, y con frecuencia hay varios de ellos.

La gran potencia de este proyecto implica también algunos de sus inconvenientes: en primer lugar, al ser un proyecto de código abierto significa que los desarrolladores contribuyen con sus programas a “tal cual”. Aunque hay sistemas de control para evitar la no-ejecución del código, es más difícil de garantizar (a excepción de la honestidad de los desarrolladores) que se ejecuta como se indica. El poder del Bioconductor también se basa en la flexibilidad de la lengua **R**. Es muy difícil para los usuarios que no son competentes **R** hacer un uso eficiente de estas bibliotecas.

A pesar de estas aparentes dificultades, Bioconductor es la herramienta elegida por muchos estadísticos y la razón principal es que, cuando uno ha sido capaz de sentirse cómodo con ella, su poder es difícil de igualar. Las instalaciones de la programación de **R** hacen posible automatizar el análisis, así como la generación de informes, por lo que es la opción de elección cuando se realizan tareas repetitivas.

El software propietario

Hay muchas herramientas comerciales disponibles para el análisis de microarrays de datos. Estos van desde pequeños programas específicos de un tipo de datos en paquetes de software grandes, como Partek Genomics Suite que es una solución completa optimizada para los cálculos eficientes y rápidos, así como para la mayoría de los datos genómicos existentes. Software comercial de microarrays tiene los pros y los contras del tradicional software comercial: Puede ser bueno, pero es caro y puede que no sea suficientemente flexible para un usuario experto que desea introducir sus propios métodos en el análisis.

1.4.2 Bases de datos de Microarrays

La diversidad de formatos y tipos de microarrays de experimentos ha hecho difícil que un formato de base de datos cualquiera se haya impuesto y no hay sistema de base de datos que se haya convertido en el “dorado–estándar”.

En efecto, existe como estado algún tipo de acuerdo sobre la información mínima sobre un experimento de microarrays que debe ser almacenada (the MIAME standard (<http://www.mged.org/Workgroups/MIAME/miame.html>) es un acrónimo de esto), pero como si fuera un tema político que el acuerdo haya sido tan corto que es más simbólico que útil.

Se pueden distinguir dos niveles en los que los sistemas de bases de datos se han desarrollado.

1) *Sistemas de bases de datos locales* El análisis de los datos de microarrays pasa por una serie de pasos en los diferentes tipos de datos, imágenes, archivos binarios, archivos de texto tienen que ser procesados. Se requiere que tener almacenados en un lugar fácilmente accesible. Algunos sistemas como BASE(<http://base.thep.lu.se/>) o caArray (<http://caarray.nci.nih.gov/>) son soluciones de gran alcance para el almacenamiento de datos y experimentos, pero su uso está lejos de ser tan extendido como el de las herramientas de software de análisis.

2) *Repositorios públicos de arrays* La comunidad biológica se ha comprometido, desde el inicio de los microarrays, que los datos de los experimentos publicados deben hacerse públicos. Esto ha creado la necesidad de repositorios de microarrays pública donde cualquier usuario puede almacenar sus datos en una forma adecuada. Al mismo tiempo se ha hecho una impresionante cantidad de datos disponibles para un nuevo análisis para cualquiera que desee hacerlo, que ofrece una riqueza sin precedentes de oportunidades cuyo poder está empezando a mostrar. Una lista de las colecciones de datos pública disponible en <http://www.ncbi.nlm.nih.gov/geo/>.

1.5 Extensiones (1): Otros tipos de microarrays

Este curso se centrará, en torno al tipo más popular de microarrays: los microarrays de expresión de ARN, diseñados para estudiar la expresión génica basada en la información sobre la cantidad de ADN que se transcribe el ARNm.

La disponibilidad de las tecnologías del genoma ha permitido desarrollar otros tipos de microarrays. Por “otros”, se puede decir que se basan en microarrays de ADN para estudiar otros problemas de microarrays de expresión o que dependen de otras sustancias como las proteínas o los hidratos de carbono. Una descripción completa de cada tipo, su uso, objetivos y análisis de los datos está fuera del alcance de este trabajo. Sin embargo, para dar un ejemplo de las similitudes

y diferencias entre los microarrays de expresión y las tecnologías relacionadas hacemos una breve reseña de los problemas que requieren de estas tecnologías alternativas y dar una breve descripción de uno de ellos, los arrays de genotipado o “de SNPs”.

1.5.1 Otros microarrays de ADN

Uno de los ejes principales de la genómica funcional es la comprensión y la curación de la enfermedad. Se sabe que muchas alteraciones genéticas subyacen anomalías y/o enfermedades. Por ejemplo:

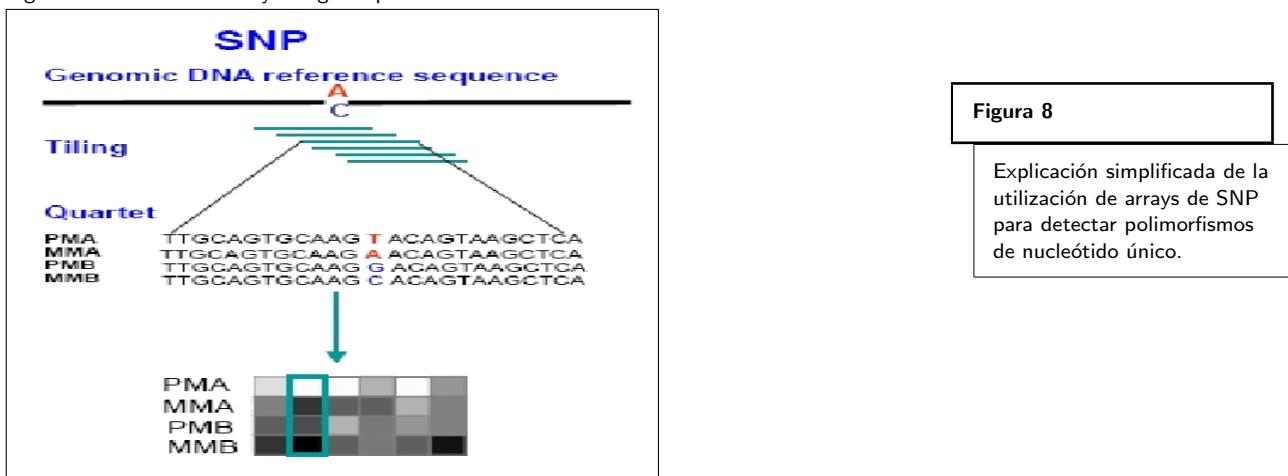
- Mutaciones puntuales –cambio de una o más bases– puede dar lugar a proteínas alteradas o cambios en el nivel de expresión.
- La pérdida de copias de genes puede reducir el nivel de expresión. Estos cambios están relacionados con la supresión de tumores.
- Ganancia de copias del gen puede aumentar el nivel de expresión y están con la activación de oncogenes relacionados.
- Metilación o de-metilación de los promotores de genes, respectivamente, pueden disminuir o aumentar el nivel de expresión. Estos también están relacionados con los oncogenes supresores de tumores.

Los diferentes tipos de microarrays se adaptan a estudiar las manifestaciones y los efectos de estas alteraciones. Los puntos planteados anteriormente pueden ser estudiados con (i) *genotipado* o SNP y (ii) *hibridación comparativa del genoma* o microarrays de ADN CGH y otros como *la metilación* o arrays de *promotores*.

Arrays de genotipado (SNPs)

El polimorfismo de un nucleótido es una forma de mutación puntual que consiste en las variaciones de pares de bases individuales que se encuentran dispersos al azar por todo el genoma. Miles de polimorfismos de un nucleótido han sido –y siguen siendo– identificados como parte de los proyectos de secuenciación del genoma. SNPs han sido altamente conservados durante la evolución y dentro de una población. Debido a esta medida de conservación el mapa de SNPs sirve como un excelente marcador genotípico para la investigación.

Los arrays SNP son un tipo de chips de ADN para detectar polimorfismos dentro de las poblaciones. Estos trabajan bajo los mismos principios básicos que arrays de expresión, pero cada sonda está diseñada para detectar las diferentes variaciones de polimorfismos de nucleótido único para cada SNP conocidos.

**Figura 8**

Explicación simplificada de la utilización de arrays de SNP para detectar polimorfismos de nucleótido único.

Los arrays SNP tienen muchas aplicaciones. Entre ellos se puede destacar:

- **Estudios basados en el linaje familiar** El ADN de los familiares afectados con una enfermedad en particular puede ser comparado con el ADN de los miembros de la misma familia que no tienen esta condición. Estos estudios permiten identificar las diferencias genéticas que pueden estar asociadas con la enfermedad.
- **Estudios de asociación a nivel poblacional** consiste en determinar las diferencias en las frecuencias de SNP en los individuos afectados y no afectados en una población. El objetivo es identificar SNPs en particular o combinaciones de SNP que difieren entre los dos grupos y por lo tanto, asociados con la enfermedad. Estos estudios requieren un gran número de muestras para representar adecuadamente a la población. Este es uno de los mejores - aplicación conocida de las matrices de SNPs que ilustra cómo se puede ayudar en la identificación de genes relacionados con enfermedades complejas.
- **Cambios del número de copias** SNPs pueden ser utilizados como etiquetas para las regiones con variabilidad del número de copias –una variante del número de copias (CNV)– es un segmento de ADN que es de 1 KB o más grande y está presente en un número variable de copias en comparación con un genoma de referencia”. La identificación de los cambios de número de copias es útil para detectar tanto las aberraciones cromosómicas como la pérdida del número de copias neutrales de heterocigosis (LOH), eventos que son característicos de muchos tipos de cáncer.

1.5.2 Otros tipos de microarrays: Proteínas o carbohidratos

Existe un amplio consenso sobre el hecho de que la información obtenida de los microarrays de ADN no es suficiente para alcanzar una completa comprensión de los procesos celulares la mayoría de los cuales son controlados por las proteínas que interactúan con otras moléculas como los hidratos de carbono a menudo implicados en importantes mecanismos biológicos como la interacción de patógenos

con el huésped, el desarrollo o la inflamación. Las microarrays de proteínas (i) y carbohidratos (ii) son dos ejemplos de la extensión del uso de estas herramientas para el análisis de alto rendimiento de diferentes tipos de moléculas. Microarrays de tejidos (iii) son un tipo diferente de extensión en la que la resta no es distintas variantes de un solo tipo de molécula, sino de un tipo de tejidos.

1.6 Extensiones (2): *NGS: Next(Now) Generation Sequencing*

A finales de la primera década del siglo XXI parece ser que la revolución de los microarrays está llegando a su fin ([35, 20]) y una nueva tecnología está inundando las revistas y congresos científicos. Se trata de la ultrasecuenciación también llamada “Next Generation Sequencing” o ”Now Generation Sequencing“.

Básicamente la ultrasecuenciación permite hacer lo mismo que la secuenciación tradicional o “Sanger”, es decir obtener la secuencia de una cadena de ADN o ARN que se ha preparado previamente para ello mediante creación de un cierto número de copias o “librerías”. La diferencia, una vez más, está en la cantidad de secuencias que es posible obtener. Mientras que la secuenciación tradicional suele poder producir XXX pares de bases por día la capacidad de los nuevos métodos es órdenes de magnitud superior lo que significa el acceso rápido y económico a la capacidad de secuenciar genomas de eucariotas en un tiempo breve -semanas o días en 2012- a un coste ínfimo ¹de secuenciar.

Este gran incremento en la capacidad de producir secuencias ha representado, de nuevo, un cambio de paradigma en la resolución de muchos problemas científicos. Hoy es posible considerar una aproximación directa al estudio de múltiples problemas (“Si quieres saber como va secuencialo”) y la secuenciación se aplica a un gran número de campos desde la metagenómica que estudia a nivel genómico la diversidad de los ecosistemas bacterianos como lagos o intestinos, al análisis de variantes estructurales en exomas o genomas -y su posible asociación con enfermedades hereditarias como el autismo o el cáncer de mama. Una variante de la secuenciación de ADN ha sido el RNA-seq que, secuenciando ADN complementario permite abordar un estudio mucho más fino de la expresión génica de la que se pueda hacer con microarrays, dado que permite cuantificar directamente el producto de la expresión -podemos secuenciar el ARN y contar los transcritos para saber cuanta expresión se ha dado- a la vez que es posible obtener información acerca de secuencias no identificadas previamente -por lo que no se habían podido poner en los microarrays.

En esta sección se presenta una visión general de la ultrasecuenciación, las tecnologías que funcionan en 2011, los problemas bioinformáticos y computacionales que aparecen y como se resuelven y algunos de los problemas que pueden abor-

¹ A principios de 2012 puede obtenerse un genoma humano en menos de un mes por unos 10.000 euros pero ya hay compañías que anuncian el genoma de 100€ en 24 horas para antes de un año.

darse con estas técnicas con un sesgo hacia los aspectos bioinformáticos y de análisis de datos que no dejan de ser el objeto de este curso.

1.6.1 Visión global de las tecnologías de secuenciación

El ADN no puede ser secuenciado de golpe, ha de ser fragmentado suficientemente, secuenciado a trozos y finalmente reensamblado. La característica básica presente, tanto en el paradigma tradicional (Sanger) como en ultrasecuenciación es que, previa fragmentación de ADN a secuenciar, cada fragmento deberá ser amplificado o clonado. Debido a este paso previo de fragmentación y amplificación se puede acuñar a las tecnologías de secuenciación en general con el término *shotgun sequencing* o *shotgun cloning*. En cierto modo mayor amplificación implica mayor precisión de lectura. Dado un fragmento concreto a secuenciar y un ciclo de proceso, se intervendrá químicamente en su colonia o grupo de cadenas clones asociada (*polony*) de una manera particular dependiendo de la tecnología concreta. La primera diferencia entre la tecnología convencional (Sanger) y las NGS es que la primera necesita que el resultado de la intervención sea diferente para cada cadena de la *polony*, mientras que las segundas todo lo contrario. Al final de este capítulo entenderemos que la precisión del resultado depende de la eficiencia conseguida en uno u otro sentido.

Desde principios de los 90, la producción de secuenciación de ADN ha sido llevada a cabo casi exclusivamente mediante implantaciones automáticas, y con escasa capacidad de procesamiento paralelo, de la bioquímica de Sanger. [4, 5].

1.6.2 De la secuenciación Sanger a la ultrasecuenciación

El método Sanger evolucionó desde su versión más artesanal hasta su versión automatizada actual. La clave principal del método de Sanger artesanal fue el uso de didesoxinucleótidos trifosfato (ddNTPs) como terminadores de cadena, motivo por el cual también se hace referencia a éste con el término *chain termination method*. Para la lectura de un solo fragmento, deben realizarse por separado, cuatro mezclas de reacción. Cada mezcla de reacción contiene los cuatro nucleótidos trifosfato (dATP, dCTP, dTTP . dGTP), ADN polimerasa I, un cebador o *primer* marcado radiactivamente (permite el acoplamiento de la polimerasa e inicio de la reacción en ese punto) y un nucleótido didesoxi ddNTP (A,C,G, o T), a una concentración baja. El nucleótido didesoxi utilizado competirá con su homólogo por incorporarse a los clones del fragmento dado, produciendo la terminación de la síntesis en el momento y lugar donde se incorpora, ya que al carecer de grupo 3'-OH no es posible el enlace fosfodiester con el siguiente nucleótido. Por este sistema, en cada mezcla de reacción se producen una serie de moléculas de ADN de nueva síntesis de diferente longitud que terminan todas en el mismo tipo de nucleótido y marcadas todas radiativamente por el extremo 5' (todas contienen en el extremo 5' el *primer* utilizado).

Las secuencias de ADN de nueva síntesis obtenidos en cada mezcla de reacción se separan por tamaños mediante electrofresis en geles verticales de acrilamida muy finos (0.5 mm de espesor) y de gran longitud (cerca de 50 cm) que permiten distinguir secuencias de ADN que se diferencian en un solo nucleótido. Los productos de cada una de las cuatro mezclas de reacción se insertan en cuatro carriles diferentes del gel.

Una vez terminada la electrofresis, el gel se pone en contacto con una película fotográfica de autorradiografía. La aparición de una banda en una posición concreta de la autoradiografía en uno de los cuatro carriles nos indica que en ese punto de la secuencia del ADN de nueva síntesis (complementario al ADN molde) está la base correspondiente al nucleótido didesoxi utilizado en la mezcla de reacción correspondiente.

Teniendo en cuenta que el ADN de nueva síntesis crece en la dirección 5' -3', si comenzamos a leer el gen por las secuencias de menos tamaño (extremo 5') y avanzamos aumentando el tamaño de las secuencias (hacia 3'), obtendremos la secuencia completa del ADN de nueva síntesis en la dirección 5' -3' del fragmento dado.

La principal diferencia entre el método enzimático de terminación de cadena y el método automático de secuenciación radica, en primer lugar, en el tipo de marcaje. En el método automático en vez de radiactividad se utiliza fluorescencia y lo habitual es realizar cuatro mezclas de reacción, cada una con nucleótido trifosfato (dTTP) marcado con un fluorocromo distinto. Este sistema permite automatizar el proceso de manera que es posible leer al mismo tiempo los ADNs de nueva síntesis producto de las cuatro mezclas de reacción.

Después de tres décadas de perfeccionamiento gradual, la bioquímica Sanger puede ser aplicada actualmente para alcanzar longitudes de lectura de hasta 1000 pares de bases ("bp") y precisiones crudas por base de hasta un 99.999% aunque con un coste de 0.5 \$ por kilobase.

1.6.3 Tecnologías de ultrasecuenciación

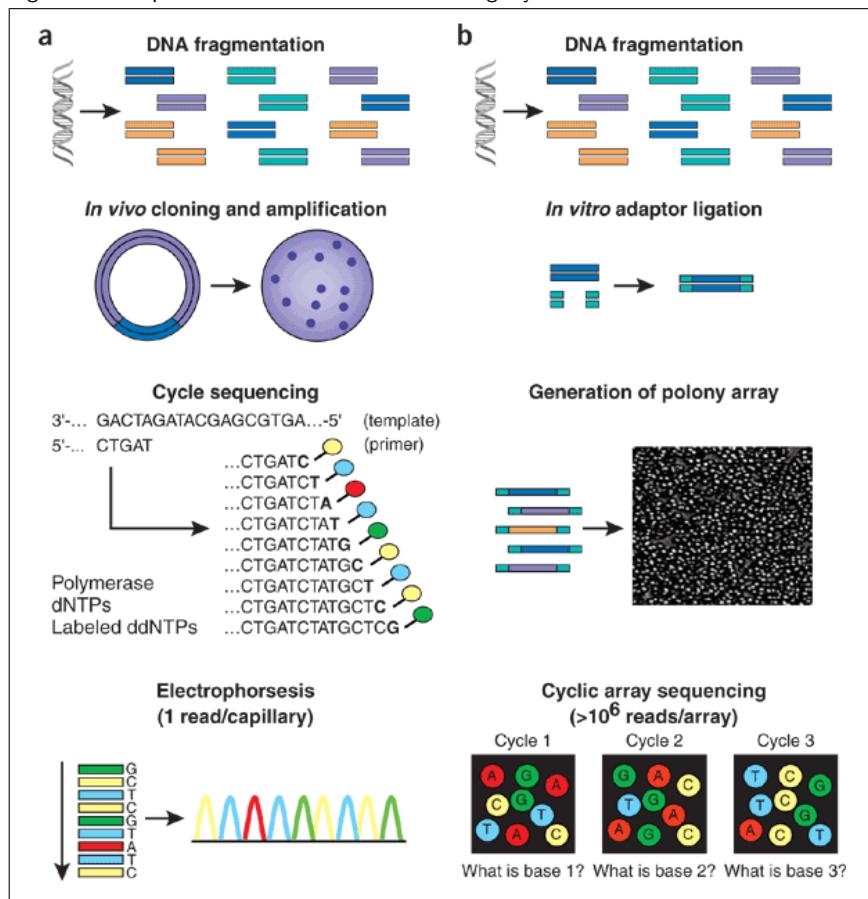
Las distintas estrategias NGS pueden ser agrupadas dentro de varias categorías. Nosotros centramos nuestro interés en el desarrollo posterior únicamente en la categoría *cyclic-array sequencing*. El resto de categorías corresponden a tecnologías en desarrollo y quedarán muy brevemente introducidas en la siguiente lista:

- *Microchip-based electrophoretic sequencing*: se trata de un intento de incremento de automatización y paralelismo de los métodos tradicionales; todavía en desarrollo, idealmente, se pretende integrar todos los aspectos del procesado de muestras, lo que reduciría sustancialmente el tiempo de proceso y el consumo de reactivos sin renunciar a la precisión del método tradicional.
- *Real-time observation of single molecules*: una de las aproximaciones en este sentido viene de la mano de Pacific Biosciences; debido a la restricción de las reacciones de iluminación acopladas a la actividad de la polimerasa a un volumen del orden del zeptolitro (10^{-21}L), están consiguiendo que la incorporación de los nucleótidos pueda ser observada con muy bajo ruido; incluye un potencial de longitud de lectura mayor que el de Sanger con una capacidad enorme de paralelismo.
- *Sequencing by hybridization* : el concepto básico de esta aproximación es que las diferencias de hibridación de ciertos fragmentos de ADN etiquetados contra un array de sondas de oligonucleótidos pueden ser usadas para identificar con precisión posiciones de variabilidad (variantes).
- *Cyclic-array sequencing*:
 - secuenciación 454 (usada en los 454 Genome Sequencers, Roche Applied Science; Basel).
 - tecnología Solexa (usada en el Illuminia (San Diego) Genome Analyzer).
 - la plataforma SOLiD (Applied Biosystems; Foster City, CA, USA).
 - Polonator (Dover/Harvard).
 - tecnología HeliScope Single Molecule Sequencer (Helicos; Cambridge, MA, USA).

Aunque las plataformas incluidas en nuestra categoría de interés (*cyclic-array sequencing*) difieren en su bioquímica, el flujo de trabajo es conceptualmente similar (Fig. 9a). Hay varias aproximaciones para la generación de los clusters de clones (*polonies*), pero todas ellas suponen que al final los derivados PCR de una única molécula de la librería acaben clusterizados en un soporte espacial. Entre ellas están las *in situ polonies* (sujeción de localización única), *bridge PCR* (sujeción a sustrato plano) y *emulsion PCR* (sujeción a gotas (*beads*) de tamaño en torno al micrómetro (10^{-6}m o *micron*)). El proceso de secuenciación (véase fig:

9b) consiste en general en alternar ciclos de irrigación enzimática con adquisición de datos basada en procesado de imagen.

Figura 9. Comparación entre la secuenciación Sanger y la ultrasecuenciación.



2. El lenguaje estadístico R

2.1 ¿Qué son R y Bioconductor?

R es un entorno de programación diseñado específicamente para trabajos estadísticos. Se trata de un proyecto de software libre, resultado de la implementación GNU del lenguaje S. R es, probablemente, uno de los lenguajes más utilizados por la comunidad estadística, siendo además muy populares en el campo de la investigación biomédica, la bioinformática y las matemáticas financieras. Se trata, además, de un programa modular, lo que significa que los usuarios pueden escribir extensiones y distribuir los códigos a otros usuarios. EL código se distribuye en forma de librerías listas para su utilización que se denominan *paquetes*. La mayor parte de la funcionalidad de R está en los paquetes, no sólo en los realizados por los desarrolladores del programa madre sino también los realizados por otros usuarios. Por su parte Bioconductor es un proyecto de código abierto para el análisis de datos de genómica, que contiene un repositorio de paquetes de R para bioinformática. Sus orígenes se remontan al otoño del 2001 con el objetivo de desarrollar e integrar software para el análisis estadístico de datos de laboratorio en biología molecular. Su principal aplicación es el análisis de microarrays.

2.1.1 Descarga e instalación de R

R se distribuye bajo la licencia GNU GPL y está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux. Además, también están disponibles los códigos fuente para la compilación de otros sistemas operativos. La página de descarga del programa R se encuentra en <http://www.r-project.org>. Se busca el programa de instalación del sistema que nos interesa y se descarga al ordenador. Se instala el programa base de R y se recomienda también la instalación de algunos paquetes. En los siguientes apartados se describe la instalación detallada en los diferentes sistemas operativos.

Instrucciones para la instalación en Windows.

Para instalar R en Windows basta con instalar el ejecutable que aparece en la web: <http://cran.es.r-project.org/bin/windows/base/>. Para ello, descargue el archivo al ordenador para posteriormente ejecutarlo. El programa de instalación se ejecuta mejor con privilegios de administrador. Acepte la licencia de uso y seleccione la carpeta de instalación. Tenga en cuenta que R requiere 1 GB de memoria disponible en el equipo para su correcto funcionamiento.

Instrucciones para la instalación en Mac OSX.

El directorio `bin / macosx` del repositorio principal de R llamado CRAN contiene un paquete estándar de Apple para instalar R llamado `R.dmg`. Una vez descargado y ejecutado, el programa instalará la versión actual de R para no-desarrolladores. **RAqua** es una versión nativa de R para Mac OS X de Darwin con un R.app Mac OS X GUI. En el interior de `bin/macosx/powerpc/contrib/x.y` hay paquetes precompilados binarios (para la versión PowerPC de Mac OS X) para ser utilizado con la RAqua correspondiente a la versión de R “x.y”. La instalación de estos paquetes está disponible a través del menú “Package” de la interfaz gráfica de usuario R.app. La página de R para Mac OS X FAQ tiene más detalles (<http://cran.r-project.org/bin/macosx/RMacOSX-FAQ.html>).

Instrucciones para la instalación en UNIX y GNU/Linux

El código siguiente es una muestra de como realizar una instalación básica en un sistema GNU/Linux desde los repositorios habituales en la distribución Debian (Ubuntu y similares):

```
sudo apt-get install r-base r-base-dev r-recommended
```

El equivalente para distribuciones de GNU/Linux basadas en Fedora/RHEL sería:

```
yum install R R-devel R-lattice
```

Instrucciones para la instalación de Bioconductor

Para instalar Bioconductor debemos tener instalado de forma correcta el programa R. La forma más sencilla de proceder es usando un script de instalación ya creado. En primer lugar, se debe descargar y ejecutar el script desde una sesión de R:

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite()
```

Si no se puede acceder a Internet directamente desde R, pueden descargarse los paquetes desde el sitio de Bioconductor en <http://www.bioconductor.org>. Si no sabe cómo descargar paquetes adicionales en el programa R puede consultar el siguiente apartado.

Instalación de paquetes adicionales en cualquier sistema

Todos los paquetes de R y Bioconductor se almacenan en Internet en lugares (servidores) accesibles libremente que se denominan *repositorios*. CRAN es el repositorio central para los paquetes de R en general, pero los paquetes Bioconductor se almacenan en un repositorio propio (<http://bioconductor.org>). Los paquetes de los repositorios de CRAN o Bioconductor se pueden instalar directamente desde R.

En primer lugar se debe seleccionar el repositorio desde R:

```
> setRepositories()
```

Esto nos da una lista de repositorios disponibles. Debemos seleccionar al menos uno y especificar:

```
> install.packages()
```

Bioconductor también contiene muchos paquetes de anotación, como KEGG y como GO. Los paquetes disponibles se pueden comprobar en <http://www.bioconductor.org/download/metadata/>. La última versión se debe seleccionar de la lista de actualizaciones. Los nombres de los paquetes son exactamente como aparecen en la lista de la página web. Un ejemplo sería el siguiente:

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite(c("KEGG", "GO", "cMAP"))
```

Se pueden instalar al mismo tiempo un número ilimitado de paquetes con esta sintaxis. Sólo se tienen que añadir los nombres de los nuevos paquetes entrecomillados y separados por comas (,).

Entre los paquetes de anotación destacan por ejemplo las anotaciones para todos los chips de la compañía Affymetrix.

Por ejemplo, para instalar toda la información relacionadas con la levadura ygs98 se debe escribir:

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite(c("ygs98", "ygs98cdf", "ygs98probe"))
```

Estos paquetes contienen las anotaciones de las sondas o “probes”(ygs98), las asignaciones entre las sondas y grupos de sondas (“probesets”) (ygs98cdf), y las secuencias de la sonda (ygs98probe).

Instalación de paquetes adicionales de archivos zip o tar.gz

Si no puede conectarse a Internet directamente desde R, se pueden descargar los paquetes en el equipo local para ser instalados posteriormente. Los paquetes para Windows son archivos .zip y los paquetes para sistemas UNIX ó Mac OSX son archivos con extensión .tar.gz. En Windows un paquete zip se puede instalar desde el menú de interfaz de usuario “Packages” usando la opción `Install package(s) from local zip files`. Para los sistemas UNIX, la instalación es un poco más complicada. Se utiliza el comando `install.packages()`. Por ejemplo, para instalar un paquete llamado mi_archivo el comando sería:

```
> install.packages(repos = NULL, Pkgs = "mi_archivo.tar.gz")
```

Antes de ejecutar el script de instalación, es necesario cambiar la ruta de R para seleccionar la carpeta donde se encuentra el paquete. Esto se hace con el comando `setwd()`, y la ruta se introduce entre comillas, por ejemplo, `setwd('/uoc/curso')`.

2.1.2 Interfaces gráficas de usuario

Existen varias interfaces gráficas de usuario en R, que van desde editores de texto integrados con R hasta interfaces gráficas del estilo apuntar y clicar. Uno de ellos es RStudio.

RStudio

RStudio es un entorno de desarrollo integrado (IDE) de R con una interfaz de usuario muy intuitiva y que tiene potentes herramientas de codificación. Está disponible para todas las plataformas incluyendo Windows, Mac OS X y GNU/Linux. Al igual que R, está disponible bajo una licencia de software libre que garantiza la libertad de compartir y modificar el software. El programa se puede descargar desde el sitio <http://www.rstudio.org/download/>.

2.2 Introducción al lenguaje R

2.2.1 Conceptos básicos del lenguaje R y su entorno

Empezar con R sin ningún conocimiento previo es una tarea bastante exigente. En este capítulo se ofrece una visión general de los fundamentos de R y de algunos de los comandos R más comunes.

Iniciar y cerrar R

R se puede iniciar haciendo clic en su ícono (Windows y Mac OS X) o escribiendo su nombre en el símbolo del sistema (GNU/Linux y UNIX). R se cierra con el comando `q()`. Cuando se cierra R, el usuario tiene la opción de guardar todo lo que ha hecho en la misma sesión. R pregunta si se desea guardar el espacio de trabajo (`Save workspace image`). Si la respuesta es sí, todos los objetos se guardan en un archivo llamado `.RData`, y el historial de comandos se guarda en un archivo llamado `.Rhistory`. Esto es muy útil ya que estos archivos pueden cargarse de nuevo en la memoria, y el análisis se puede continuar. En Windows las ventanas de estos archivos se pueden cargar desde el menú `File -> load workspace and File -> load history`. En UNIX, GNU/Linux y Mac OS X se pueden cargar los archivos con el siguiente código:

```
> load(".RData")
```

Ayuda en R

R viene con archivos de ayuda integrados. Se puede acceder directamente desde R o usando un navegador web. El navegador web es especialmente adecuado para la búsqueda de posibles comandos y ejemplos de utilización. El acceso directo desde el entorno es más rápido y se hace con la siguiente instrucción:

```
> help.start()
```

Esto abre una nueva ventana del navegador con los ítems siguientes: una introducción a R, conceptos básicos, manuales y paquetes. También aparece una descripción detallada de comandos y conjuntos de datos en cada paquete.

Por otro lado, dispone de un motor de búsqueda que puede ser utilizado para buscar comandos y conjuntos de datos

Para acceder directamente desde R a la ayuda de comandos, el nombre del comando se escribe entre paréntesis:

```
> help(sum)
```

Al ejecutar este comando, se abre la página de ayuda en R. Por lo general en Windows se abre una nueva ventana para la página de ayuda. En UNIX, en cambio, la página de ayuda se muestra en el editor.

La disposición general de la página de ayuda es la siguiente: la primera línea da el nombre del comando y el paquete donde se puede encontrar. En segundo

lugar, encontraremos el nombre de la función o el comando. Después del nombre viene una descripción breve y general sobre el uso del comando. Por último se especificarán los argumentos que puede llevar asociado el comando. Al final de la página de ayuda se encuentran algunos ejemplos de aplicación de la función.

Asignación de datos

La asignación se realiza a través del operador “`<-`” que se compone del signo menor unido al signo menos. La asignación se utiliza para guardar información en algún objeto llamado en la memoria R. La asignación elimina el objeto que tenga el mismo nombre.

Para guardar un solo número de un objeto denominado `number` se hace de la siguiente manera:

```
> number<-2
```

Los nombres de los objetos tienen unas normas. No pueden empezar con un número o con un nombre que contenga caracteres especiales, tales como ä, ñ, etc. Si el nombre de la variable es largo, es preferible separar las partes con un punto (.), no con espacios en blanco (), un signo menos (-) o un guión bajo (_). También es mejor evitar el uso de nombres de objetos iguales a nombres de instrucciones.

2.2.2 Tipos de objetos en R

A continuación se presentan los diferentes tipos de objetos que encontramos en el programa R. Es muy importante conocer el tipo de objeto con el que estamos trabajando de cara a usar los instrucciones ya que no todos los instrucciones y funciones están disponibles para todos los objetos. Algunas instrucciones estadísticas pueden generar listas muy largas de objetos. Para obtener una lista de todos los elementos contenidos en el objeto podemos utilizar `str()` de la forma siguiente:

```
> dat3<-matrix(c(1,2,3,11,12,13),nrow=2,ncol=3,dimnames=list(c(),  
+ c("col1","col2","col3")))  
> class(dat3)
```

```
[1] "matrix"
```

```
> str(dat3)
```

```

num [1:2, 1:3] 1 2 3 11 12 13
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:3] "col1" "col2" "col3"

```

En este ejemplo el comando **str()** ofrece, en una primera línea, una descripción detallada de una matriz que contiene 3 columnas y 2 filas. En la segunda línea dice que hay nombres para las dimensiones de la matriz, y que estos nombres configuran una lista. Las columnas de la matriz tienen los nombres col1,col2 y col3 pero las filas no contienen los nombres y por consiguiente los nombres aparecen como NULL (sin datos).

Vectores

Un vector es una lista ordenada de números o cadenas. La forma de definirlo es mediante **c()** indicando entre los paréntesis los elementos del vector separados por comas. Por ejemplo:

```
> x <- c(1, 2, 3); y <- c("a", "b", "Hola")
```

x es un vector con tres valores numéricos e **y** un vector con tres cadenas de caracteres.

```
> z1 <- c(TRUE, TRUE, FALSE)
```

z es un vector con tres valores lógicos (cierto, cierto y falso).

Una de las funciones útiles al trabajar con vectores es **length()**, que calcula la longitud (número de valores) de un vector:

```

> a<-c(2.258, 3.458, 4.897, 1.241, 2.741)
> length(a)

```

```
[1] 5
```

Factor

Un factor es un vector de elementos categóricos (o categorías) que suelen ser utilizados cuando es preciso definir grupos por ejemplo en los modelos de análisis de la varianza.

Un vector cuyos elementos sean números o cadenas puede convertirse fácilmente en un factor utilizando el comando `as.factor()`:

```
> xf<-as.factor(c(1,1,1,2,2,2,3,3,3))
> xf
```

```
[1] 1 1 1 2 2 2 3 3 3
Levels: 1 2 3
```

Por defecto, `as.factor()` convierte el vector a un factor con tantos niveles distintos como valores distintos estaban presentes en el vector.

Matriz

Una matriz es una tabla de `n` filas y `m` columnas. Todas las columnas de una matriz deben contener información del mismo tipo. Por lo tanto podemos tener matrices numéricas, o con datos del tipo cadena pero no se pueden mezclar en la misma matriz los números y las cadenas . Por ejemplo:

```
> dat2<-matrix(c(0,1,1,1,0,0,1,1,1,1,0,0),
+                 nrow=6,ncol=2,dimnames=list(c(),c("x","y")))
> class(dat2)

[1] "matrix"
```

Podemos acceder a todos los valores de una matriz mediante un subíndice. El subíndice indica la fila y la columna de la observación a la que queremos tener acceso dentro de la matriz:

```
> # La primera fila y primera columna
> dat2[1,1]
```

```
x
0
```

```
> # Solo la primera fila
> dat2[1,]
```

```
x y
0 1
```

```
> # Solo la primera columna  
> dat2[,1]
```

```
[1] 0 1 1 1 0 0
```

```
> # Las filas de 1 a 3  
> dat2[1:3,]
```

```
  x  y  
[1,] 0 1  
[2,] 1 1  
[3,] 1 1
```

Las dimensiones de una matriz se pueden comprobar con el comando `dim()` que imprime dos números. El primero indica el número de filas y el segundo el el número de columnas:

```
> dim(dat2)
```

```
[1] 6 2
```

Data frame

Un `data frame` es una matriz, donde las diferentes columnas pueden contener información de diferentes tipos. En contraste con las matrices, un `data frame` puede contener columnas de números junto a columnas de cadenas. Es el objeto que se utiliza para representar una matriz de datos donde las columnas son las variables y en las filas se encuentran los individuos o elementos sobre los que se han medido las variables. En un `data frame` las columnas tienen un nombre o etiqueta (el de la variable) y también se les puede dar un nombre a las filas (identificador del elemento). La manera de acceder a un valor concreto se puede hacer de forma idéntica a como se hace con las matrices. Los nombres de las columnas y la filas se pueden encontrar usando los instrucciones `colnames()` y `rownames()`, respectivamente.

Podemos crear un `data frame` a partir de 2 vectores:

```
> x<-c(0,1,0,0,1,0,0,1,0)  
> y<-c(0,0,1,0,0,1,0,0,1)  
> dat2<-data.frame(x, y)
```

Los nombres de los valores del **data frame** son muy fáciles de cambiar. Por ejemplo:

```
> dat2
```

x	y
1	0
2	1
3	0
4	0
5	1
6	0
7	0
8	1
9	0

```
> names(dat2)
```

```
[1] "x" "y"
```

```
> row.names(dat2)
```

```
[1] "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

```
> # Establecemos los nombres de las filas  
> # Para nombrar cada fila usamos letras  
> l<-letters[1:9]  
> l
```

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i"
```

```
> row.names(dat2)<-l  
> row.names(dat2)
```

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i"
```

Además de los subíndices, podemos acceder a cada columna de un **data frame** usando su nombre. El nombre se escribe después del nombre del **data frame** seguido de un signo de dólar (\$):

```
> dat2$x
```

```
[1] 0 1 0 0 1 0 0 1 0
```

Una matriz existente se puede convertir en un **data frame** con un comando de conversión específico como:

```
> dat2 <- as.data.frame (dat2)
```

Lista

Una lista es un “vector generalizado” es decir un contenedor diseñado para almacenar objetos de tipos potencialmente distintos (y, por lo tanto, pueden ser otras listas)

```
> una.lista <- list(un.vector = 1:10,      # primer elemento un vector
+                     una.palabra = "Hola",    # segundo: cadena de caracteres
+                     una.matriz = matrix(rnorm(20), ncol = 5),
+                                         # el tercer elemento: es una matriz
+                     otra.lista = list(a = 5, b = factor(c("a", "b"))))
+ )                                # cuarto: lista de 2 elementos
```

Instrucciones de R

Los comentarios se escriben a continuación del símbolo `#`. Si el comentario ocupa más de una línea en cada nueva línea se debe escribir el símbolo que lo caracteriza. Las instrucciones de R se escriben todas en minúscula o en mayúscula y llevan un paréntesis inmediatamente después del nombre. Por ejemplo, consideremos la instrucción `mean()` que permite calcular la media aritmética de un objeto (`x`). Por ejemplo, la edad de diez personas se pueden almacenar en una variable que llamaremos `edad`. Si queremos buscar la media de edad del grupo se puede calcular con:

```
> mean(edad)
```

Cada instrucción se compone de tres partes:

- nombre del instrucción
- paréntesis

- argumento(s) que irá dentro del paréntesis

No siempre es fácil adivinar qué argumentos tiene cada instrucción. Si queremos conocer los argumentos de nuestra función, lo mejor es consultar la página de ayuda de nuestra instrucción en concreto. Además, buscar las instrucciones en la página de ayuda es muy útil para probar los ejemplos.

En general, los valores numéricos o lógicos se escriben sin comillas. En el caso de los valores en las cadenas de carácter estos se deben escribir entre comillas. Por ejemplo, el comando `mean()` no usará ningún valor cadena y todos los argumentos que se especifican sin las comillas:

```
> mean(age, na.rm=T, trim=0.1)
```

Las instrucciones se pueden escribir una tras otra en la misma línea (anidadadas), siempre y cuando el número de paréntesis sea el correcto. Por ejemplo, si queremos quitar los valores perdidos –a los que hemos asignado un “NA” dentro de un conjunto de edades para poder calcular la media podemos escribir:

```
> # Elimina los valores perdidos de "age" y crea age2
> age2<- na.omit(age)
> # Calcula la media de "age2" (no missing values)
> mean(age2)
> # El mismo proceso pero en la misma linea
> mean(na.omit(age))
```

El entorno de R

El entorno de R tiene algunas funciones útiles. Se pueden ver instrucciones anteriores con las teclas de flecha arriba y abajo. Esta funcionalidad es especialmente útil cuando hay un comando que se ha escrito de forma incorrecta. Una vez se usa la tecla hacia arriba, vuelve a la última orden, que puede ser editada sin necesidad de volver a escribir todo el comando. También es posible obtener una lista de todos los instrucciones ejecutados en R utilizando el historial de instrucciones `history()`. Por defecto R proporciona una lista con un par de docenas de instrucciones. Para ampliar la lista se puede usar el argumento `max.show`. Por ejemplo, el comando siguiente muestra las últimas 100 instrucciones:

```
> history(max.show=100)
```

Los paquetes en R

En R, se pueden instalar los paquetes adicionales tal y como se describe en el capítulo de instalación detallado más arriba. Los paquetes instalados no se cargan en R por defecto cada vez que se inicia. Es el usuario quien debe cargar el paquete en la memoria utilizando la instrucción `library()`. Por ejemplo, para cargar el paquete (`rpart`) la instrucción será:

```
> library(rpart)
```

Para obtener una lista de todos los paquetes cargados en la memoria, se utiliza `sessionInfo()`:

```
> sessionInfo()
```

Si el paquete no se carga en la memoria, los instrucciones o conjuntos de datos contenidos en el paquete no puede ser utilizados y no se encontrará documentación para implementar las funciones. Por ejemplo, la ayuda `rpart()` daría el siguiente mensaje de error:

```
No documentation for 'rpart' in specified packages and libraries:  
you could try 'help.search("rpart")'
```

Y esto significa que no tiene información sobre la función `rpart` en el paquete especificado o en la biblioteca y pedirá que el usuario busque la función en la ayuda de la biblioteca.

A veces dos paquetes contienen funciones con exactamente el mismo nombre. Esto puede provocar la advertencia siguiente:

```
Attaching package: 'rpart'  
The following object(s) are masked \_by\_\_.GlobalEnv :  
rpart
```

Esta advertencia significa simplemente que el comando `rpart()` no se puede utilizar, porque ya hay otro comando con el mismo nombre en la memoria. Esto puede solucionarse eliminando el paquete conflictivo de la memoria mediante el comando `detach()`:

```
> detach(package:rpart)
```

Cambiar el directorio de trabajo

Es una buena práctica dedicar una sola carpeta para contener todos los ficheros relacionados con un único conjunto de datos. Cuando se empieza a trabajar con un nuevo conjunto de datos, se deben copiar en una carpeta nueva. Antes de la importación de datos, R debe decidir dónde residirán los datos. En Windows podemos escoger el directorio de trabajo usando el menú **File -> Change Dir**. Esto abre una ventana nueva donde el usuario puede escoger la carpeta correcta. En UNIX, GNU/Linux y Mac OSX, el usuario debe especificar la ruta de la carpeta de datos con el comando **setwd()**. Por ejemplo, para establecer la ruta a una carpeta de datos, el usuario puede ejecutar un comando de esta forma:

```
> setwd("/home/uoc/doc/timeseries")
```

A veces es útil para verificar el directorio de trabajo actual. El camino hacia el directorio de trabajo se puede comprobar con el comando **getwd()**:

```
> getwd()
```

Los archivos que residen en la carpeta actual se pueden listar usando el siguiente comando **dir()**:

```
> dir()
```

2.2.3 Uso de R como calculadora

R contiene una colección muy completa de funciones matemáticas y aritméticas. En R se utiliza el orden estándar de precedencia en las operaciones, por lo cual se calcularán primero las potencias y las raíces, seguido de la multiplicación y la división y, por último, las sumas y las restas.

Operaciones aritméticas

Las sumas y restas se hacen utilizando los operadores **+** y **-**:

```
> 8+3
```

```
[1] 11
```

```
> 8-3
```

```
[1] 5
```

La multiplicación se realiza mediante un asterisco (*):

```
> 4*6
```

```
[1] 24
```

La división utiliza la barra (/):

```
> 16/4
```

```
[1] 4
```

Las potencias usan el acento circunflejo: ^

```
> 3^3
```

```
[1] 27
```

Para la raíz cuadrada, se usa el comando **sqrt**:

```
> sqrt(9)
```

```
[1] 3
```

Funciones matemáticas

Además de las funciones aritméticas, en R podemos encontrar toda clase de funciones (instrucciones) matemáticas disponibles. Las funciones más importantes son los logaritmos y exponentes.

Los logaritmos se calculan utilizando el comando **log()**:

```
> log(4)
```

```
[1] 1.386294
```

El comando `log()` es el logaritmo en base 2.718282, es decir el logaritmo neperiano. Los logaritmos en base 10 se pueden calcular utilizando la función `log10()` y el logaritmo en base 2 con `log2()`. El comando `exp()` calcula un exponente:

```
> exp(2)
```

```
[1] 7.389056
```

Funciones lógicas

Las pruebas lógicas para comparar la igualdad de dos objetos se pueden hacer usando el operador `==`, y el resultado es una expresión lógica que puede ser verdadera (`TRUE`) o falsa (`FALSE`):

```
> 8==8
```

```
[1] TRUE
```

Otros posibles operadores son `>`, `<`, `>=`, `<=` y `!=` que corresponden a mayor que, menor que, mayor o igual, menor o igual o desigual. Por ejemplo al evaluar las expresiones `3 < 5` y `3 > 5` se obtendrá `TRUE` y `FALSE` respectivamente.

```
> 3<5
```

```
[1] TRUE
```

```
> 3>5
```

```
[1] FALSE
```

Número de decimales

El programa R realiza los cálculos con más decimales de los que se muestran en la salida. Por lo tanto, el número de decimales en la salida se puede limitar especificando el número de decimales que queremos con el comando `round()`:

```
> round(exp(2), digits=2)
```

```
[1] 7.39
```

2.2.4 Entrada y salida de datos

En R podemos leer los datos de varias maneras. Para la entrada de datos, la forma más sencilla es manualmente utilizando las funciones proporcionadas por el mismo programa. Otra, menos laboriosa y que sirve para grandes conjuntos de datos, es leer los datos de una tabla que se encuentra en un fichero externo. La tabla se puede crear en cualquier editor de hojas de cálculo, como Microsoft Excel, y guardar en un formato de texto delimitado por espacios o tabulaciones (ficheros .txt) o separado por comas (ficheros .csv). Por defecto en R el delimitador de decimales para los valores numéricos es el punto (.). La coma (,) es un delimitador de las listas. Por lo tanto, todos los valores numéricos que se importan en R deben ser evaluados por si contienen sólo números o números y puntos. De lo contrario los valores se leen en forma de texto.

Escritura de datos tabulares

Los datos tabulares se pueden escribir en un archivo usando el comando `write.table()`. El comando toma como argumentos el nombre del objeto a ser escrito, el nombre del archivo entre comillas y el tipo de separador de archivo (en este caso, tab). Por ejemplo:

```
> # Creamos un data.frame  
> nombres <- sample(LETTERS, 10)  
> numeros <- runif(10)  
> ab <- data.frame(nombres, numeros)  
> write.table(ab, file="ab.txt", sep="t")
```

Lectura de datos tabulares

Los datos tabulares, tales como los archivos de texto delimitado por tabuladores, se pueden leer fácilmente en R mediante el comando `read.table()`. Se debe introducir el nombre del archivo (entre comillas), un valor lógico que indica si las columnas de la tabla tienen títulos, el separador utilizado en el archivo y, opcionalmente, el número de la columna con los nombres de las filas:

```
> dat<-read.table("ab.txt", header=TRUE, sep="t", row.names=1)  
> dat
```

En este caso, un archivo de texto delimitado por tabuladores que contiene dos columnas denominadas a y b se leerá en un objeto dat. Si el separador decimal es diferente al punto (.), se puede especificar en el comando `read.table()` con el argumento `dec`. Por ejemplo, el siguiente comando leería un archivo de texto separado por (!):

```
> dat<-read.table("ab.txt", header=TRUE, sep="t", row.names=1,  
+ dec="/")
```

Guardar el resultado en un archivo

Las salidas generadas por instrucciones de R se pueden guardar en un archivo. Antes de ejecutar los instrucciones, aparece el comando `sink()` y como argumento el nombre de un archivo para guardar el resultado. Después de ejecutar los instrucciones, el comando `sink()` aparece de nuevo, y ningún resultado más se guarda. Por ejemplo:

```
> sink("result.txt")  
> print(ab)  
> print(summary(ab))  
> sink()
```

2.2.5 Operaciones con vectores

Todas las instrucciones de los que hemos hablado anteriormente son aplicables a los vectores. Cuando se ejecuta un comando sobre un vector, la función se aplica en cada elemento del vector de forma individual. También podemos encontrar instrucciones que se usan sólo para los vectores

Operadores básicos en los vectores

Podemos usar los operadores básicos para los cálculos con vectores:

```
> x<-c(3,1,2,0,1,3,3,1,0)  
> y<-c(4,0,1,0,3,2,5,3,2)  
> z<-c(0,0,1,0,1,2,0,1,2)  
> x;y
```

```
[1] 3 1 2 0 1 3 3 1 0
```

```
[1] 4 0 1 0 3 2 5 3 2
```

```
> x+y
```

```
[1] 7 1 3 0 4 5 8 4 2
```

Operadores matemáticos en los vectores

Los instrucciones más utilizados para los vectores con elementos numéricos son `sum()` y `mean()` que calculan una suma o una media aritmética de los valores del vector:

```
> sum(x)
```

```
[1] 14
```

```
> mean(x)
```

```
[1] 1.555556
```

Los instrucciones `min()`, `max()` y `range()` nos calculan el mínimo, el máximo y el rango de los números en un vector:

```
> min(x)
```

```
[1] 0
```

```
> max(x)
```

```
[1] 3
```

```
> range(x)
```

```
[1] 0 3
```

Podemos obtener el resumen de las estadísticas básicas de un vector de datos con:

```
> summary(x)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	1.000	1.000	1.556	3.000	3.000

```
> summary(y)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	1.000	2.000	2.222	3.000	5.000

Para el cálculo de la desviación estándar, la varianza y la correlación podemos usar:

```
> sd(x)
```

```
[1] 1.236033
```

```
> var(x)
```

```
[1] 1.527778
```

```
> cor(x,y)
```

```
[1] 0.5828083
```

2.2.6 Manipulación de los datos

Generación de secuencias numéricas

La forma más sencilla de generar una secuencia de números es mediante el uso de los dos puntos (:). Cuando dos puntos separan dos números se genera una secuencia de números. Por ejemplo:

```
> 1:6
```

```
[1] 1 2 3 4 5 6
```

Para generar secuencias más complicadas podemos usar el comando `seq()`. Este comando necesita tres argumentos. El primero indica el comienzo de la secuencia, el segundo el final de la secuencia, y el tercero el incremento o decremento a usar. Por ejemplo:

```
> seq(1, 6, 0.5)
```

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
```

```
> seq(6, 1, -0.5)
```

```
[1] 6.0 5.5 5.0 4.5 4.0 3.5 3.0 2.5 2.0 1.5 1.0
```

Generar secuencias repetidas

Una secuencia de repetición se puede generar con facilidad con el comando `rep()`.

Una secuencia de repetición es una manera fácil de generar nuevas variables para modelos estadísticos. La función `rep()` usa dos argumentos, qué valor o valores queremos que se repita y cuantas veces se debe repetir:

```
> rep(1, 5)      # Repetir 5 veces el n\'umero 1
```

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

```
> rep("UOC", 5)    # Repetir 5 veces la cadena UOC
```

```
[1] "UOC" "UOC" "UOC" "UOC" "UOC"
```

```
> rep(1:3, 2)      # Repetir los n\'umeros del 1 al 3, dos veces
```

```
[1] 1 2 3 1 2 3
```

```
> rep(1:3, each=2) # Repetir los n\'umeros del 1 al 3, cada uno dos veces
```

```
[1] 1 1 2 2 3 3
```

Recodificar en R

La manera más sencilla de recodificar en R es utilizando el comando `ifelse()`.

Esta función tiene tres argumentos, una comparación lógica, el valor a devolver si la comparación es cierta y el valor a devolver si la comparación es falsa.

```
> m<-c(0,1,0,11,0,11,0,0,1)
> # Si es igual a 1, devuelve 1,
```

```
> # en otro caso devuelve 0
> x<-ifelse(m==1, 1, 0)
> # Si es igual a 11, devuelve 1,
> # en otro caso devuelve 0
> y<-ifelse(m==11, 1, 0)
> x; y
```

```
[1] 0 1 0 0 0 0 0 0 1
```

```
[1] 0 0 0 1 0 1 0 0 0
```

Es una buena práctica, para comprobar que nuestro programa funciona y hemos recodificado correctamente, hacer uso del comando `table()` que calcula una tabla de contingencia de dos vectores:

```
> table(x, y)
```

```
      y
x 0 1
 0 5 2
 1 2 0
```

Combinación de tablas

La fusión de dos tablas se hace con el comando `merge()`. Este comando tiene dos argumentos obligatorios, los nombres de las tablas. A menudo es necesario especificar la columna por la que queremos combinar las tablas. Las columnas se especifica mediante opciones `by.x` y `by.y`. Por ejemplo, la fusión de dos tablas podrían proceder de la siguiente manera:

```
> p<-data.frame (y=1:5)
> # Nombramos las filas con algunas letras
> row.names(p)<-c(letters[1:2], letters[7:9])
> p
```

```
      y
a 1
b 2
g 3
h 4
i 5
```

```
> r<-data.frame(x=6:10)
> # Nombramos las filas con algunas letras
> row.names(r)<-letters[1:5]
> r
```

```
x
a 6
b 7
c 8
d 9
e 10
```

```
> # La combinaci\'on se hace usando el nombre de las filas
> merge(p, r, by.x="row.names", by.y="row.names")
```

```
Row.names y x
1      a 1 6
2      b 2 7
```

Trasposición de matrices

En ocasiones la matriz de datos necesita dar la vuelta de forma que las columnas se convierten en filas y filas se convierten en columnas. Esto resulta especialmente útil cuando determinados modelos estadísticos se aplican a datos de microarrays. Las herramientas estadísticas clásicas asumen que las columnas contienen las variables y las filas las observaciones individuales para la variables. En los datos de microarrays esto no suele así, porque los chips individuales constituyen las columnas. La trasposición se realiza mediante el comando `t()`:

```
> # tabla simple de dos vectores
> dat2<-cbind(x,y)
> dat2
```

```
x y
[1,] 0 0
[2,] 1 0
[3,] 0 0
[4,] 0 1
[5,] 0 0
[6,] 0 1
[7,] 0 0
[8,] 0 0
[9,] 1 0
```

```
> dat3<-t(dat2)
> dat3

 [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
x     0     1     0     0     0     0     0     0     1
y     0     0     0     1     0     1     0     0     0
```

Clasificar y ordenar

Clasificar y ordenar vectores es una de las operaciones básicas y más útiles. Ordenar un vector, ya sea en orden ascendente o descendente se realiza con el comando `sort()`. El comando `order()` devuelve un vector de permutación que ordenará el vector original en orden ascendente. Un ejemplo de esto es:

```
> i<-data.frame(x=round(abs(c(rnorm(5)*10))), y=c(1:5))
> # Nombrar filas con letras
> row.names(i)<-letters[1:5]
> i
```

	x	y
a	1	1
b	4	2
c	26	3
d	1	4
e	12	5

```
> # Devuelve los valores en orden ascendente
> sort(i$x)

[1] 1 1 4 12 26
```

```
> # Devuelve un vector con los valores pueden ser ordenados
> order(i$x)
```

```
[1] 1 4 2 5 3
```

El resultado del comando `order()` es un poco más difícil de entender. El resultado del vector se ordenará en orden asecendente según los valores del vector original. En el ejemplo anterior, el cuarto valor (2) del vector original es el primer valor del vector resultante. El comando `order()` se puede utilizar para ordenar tablas completas. Para ordenarlas se utilizan subíndices de la siguiente manera:

```
> # Ordenar la tabla i de acuerdo con x
> i[order(i$x),]
```

x	y
a	1 1
d	1 4
b	4 2
e	12 5
c	26 3

Ordenar una tabla requiere ser prudente. Como el comando `order()` genera una permutación del vector, las filas de la tabla original se muestran en el orden indicado por el vector de permutación, y se genera la nueva tabla. Esta nueva tabla se ordena de forma ascendente de acuerdo a los valores del vector especificados en el comando `order()`.

Los valores perdidos

Los datos biológicos a menudo contienen valores perdidos. Estos se indican en R con la cadena de texto NA. Los valores perdidos complican muchos análisis, como los clústers jerárquicos o incluso el cálculo de la media aritmética de un vector.

Hay dos soluciones sencillas para el problema de valores perdidos: la eliminación y la imputación. Si los valores perdidos se eliminan de los datos, todas las filas (observaciones) que contienen al menos un valor perdido son eliminadas del conjunto de datos. La eliminación se realiza mediante el comando `na.omit()`:

```
> # Creacion un vector con un valor perdido
> vwm<-c(1,2,3,NA,5)
> # Eliminacion de valores perdidos
> vwom<-na.omit(vwm)
> vwom

[1] 1 2 3 5
attr(,"na.action")
[1] 4
attr(,"class")
[1] "omit"

> # Calculo de una media del vector
> # Si existen valores perdidos, esto da como resultado NA
> mean(vwm)
```

```
[1] NA
```

```
> # Sin valores perdidos se obtiene la media
> mean(vwom)
```

```
[1] 2.75
```

El comando `na.omit()` funciona de manera similar para las matrices y para las data frames, pero en lugar de valores individuales, se eliminan las filas enteras.

Los valores perdidos también se puede sustituir imputando un valor para todas las observaciones que faltan. La función `impute()` está disponible en la librería `e1071`:

```
> if (!require(e1071)) install.packages("e1071")
> library(e1071)
```

Hay dos maneras de imputación. Los valores perdidos pueden ser reemplazados, ya sea con la media o la mediana de otras observaciones en la misma variable. Tenga en cuenta que la imputación requiere una matriz o un dataframe como entrada.

```
> # Creando una matriz
> v<-matrix(c(1,2,3,4,11,16:20), ncol = 2)
> v
```

```
[,1] [,2]
[1,]    1   16
[2,]    2   17
[3,]    3   18
[4,]    4   19
[5,]   11   20
```

```
> v[1,1]<-NA
> v2<-impute(v, what=c("mean"))
> v2
```

```
[,1] [,2]
[1,]    5   16
[2,]    2   17
[3,]    3   18
```

```
[4,]    4   19
[5,]   11   20
```

```
> v2<-impute(v, what=c("median"))
> v2
```

```
[,1] [,2]
[1,] 3.5 16
[2,] 2.0 17
[3,] 3.0 18
[4,] 4.0 19
[5,] 11.0 20
```

2.2.7 Bucles y funciones condicionales

Los bucles son estructuras de control que permiten ejecutar una misma orden o un mismo comando en varias ocasiones. La instrucción más habitual para generar bucles que se ejecutan un número fijo de veces es la que utiliza la instrucción **for()**.

Las funciones condicionales permiten ejecutar un comando o grupo de comandos sólo si se cumple una determinada condición descrita por el usuario. La ejecución condicional de R se genera con los instrucciones **if()** y **if()...else**.

for

El bucle for requiere un índice y el número de veces que queremos repetir la misma orden. El índice puede ser cualquier objeto de R, aunque lo mejor es evitar los nombres de objetos ya existentes y los instrucciones. Las líneas que pertenecen al bucle se escriben entre llaves. Por ejemplo, un bucle que calcula 2^x para valores de x que van desde 1 hasta 5, se escribe como:

```
> # En este ejemplo i es el indice
> # que se repite 5 veces
> for(i in 1:5) {print(2^i)}
```

```
[1] 2
[1] 4
[1] 8
[1] 16
[1] 32
```

```
if
```

La instrucción `if()` contiene entre los paréntesis una operación lógica. Si la evaluación de la comparación lógica, da como resultados que es verdadera, las instrucciones que van entre llaves se ejecutan. En el ejemplo anterior, si sólo se desea realizar el cálculo de 2^x si una variable k es igual a 1, el código será:

```
> k<-1  
> if (k ==1) {for (i in 1:5) {print (2^i)}}
```

```
[1] 2  
[1] 4  
[1] 8  
[1] 16  
[1] 32
```

Se puede probar la ejecución condicional en el ejemplo anterior, cambiando el valor de k para, por ejemplo: 2,3...

```
if...else
```

A veces nos conviene ejecutar el comando si la condición descrita es falsa o no se cumple. Por ejemplo, en el programa siguiente 2^2 se calcula sólo si k es igual a 1 en caso contrario se calcula 3^2 :

```
> k<-1  
> if(k==1){print(2^2)} else {print(3^2)}
```

```
[1] 4
```

Se puede implementar el mismo procedimiento utilizando dos condiciones creadas con if:

```
> if (k==1) {print(2^2)}
```

```
[1] 4
```

```
> if (k!=1) {print(3^2)}
```

Nota: El último ejemplo utiliza un operador (`!=`) que significa desigualdad.

A veces, necesitamos realizar una operación lógica que contiene varias comparaciones a la vez. Para implementar varias condiciones podemos usar los operadores | y &. El primer operador significa "o" y el segundo, "y". Estos son los operadores lógicos, y se puede utilizar en cualquier orden. Por ejemplo, en el caso anterior, si se desea calcular 2^2 cuando $k=1$ o 2 , y 3^2 en los otros casos, el código sería:

```
> if(k==1 | k==2) {print (2^2)}
```

```
[1] 4
```

```
> if(k> 2) {print (3^2)}
```

2.2.8 Gráficos en R

El lenguaje R tiene capacidad de generar gráficos de muy buena calidad.

El comando plot

`plot()` es el comando general para crear gráficos de primer orden. Se puede utilizar para visualizar diferentes tipos de objetos. La imagen producida por este comando es típicamente un diagrama de dispersión. Se necesitan dos vectores para producir un diagrama de dispersión:

```
> # Generamos dos vectores aleatorios
> # con 100 valores de una distribuci\'on normal
> x<-rnorm(100)
> y<-rnorm(100)
> # Representaci\'on de los valores en un gr\'afico de dispersi\'on (scatterplot)
> plot(x, y)
```

Si se aplica la función `plot()` a una tabla (matriz o data frame), se realiza un gráfico de las variables 2 a 2. Por ejemplo:

```
> # Creaci\'on de un data frame con cuatro columnas
> # llamadas a, b, c, y d
> dat <-data.frame (a=rnorm(100), b=rnorm(100), c=rnorm(100), d=rnorm (100))
> plot(dat)
```

Figura 10

Scatterplot. Un diagrama de dispersión sirve para mostrar los valores de dos variables cuantitativas relacionadas

Figura 10. Representación gráfica de un Scatterplot

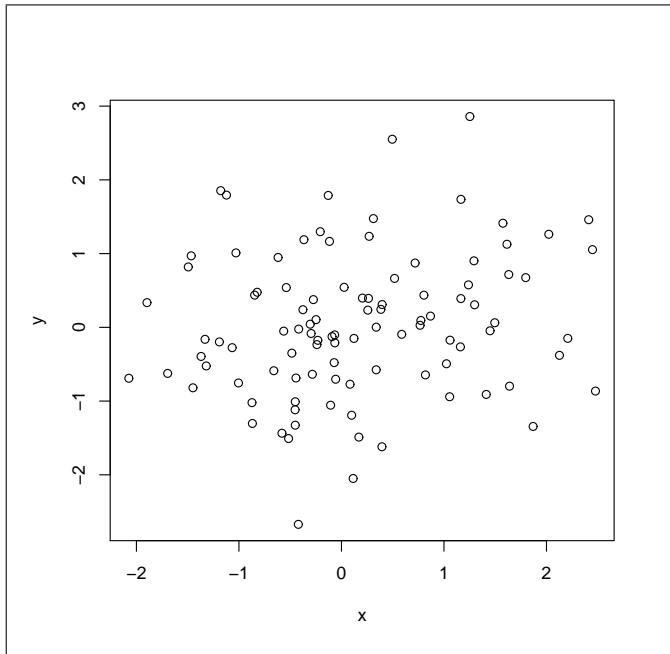
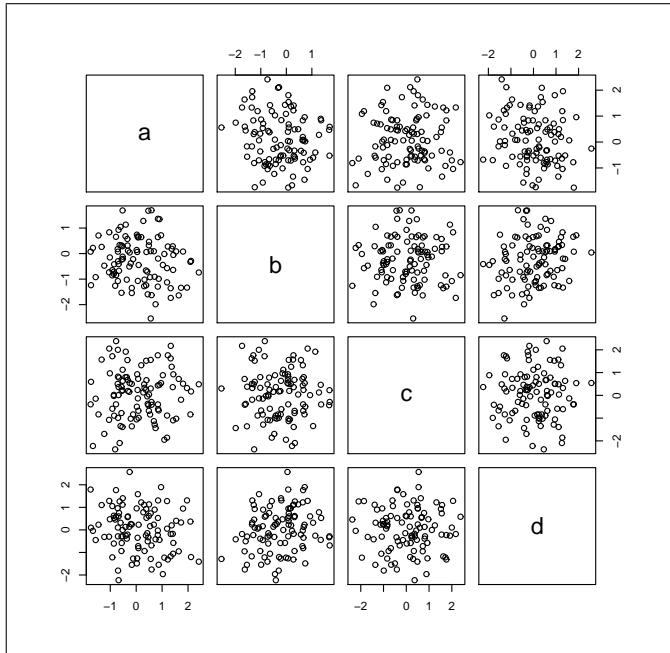


Figura 18. Representación gráfica de un conjunto de Scatterplots



El comando `plot()` contiene otros argumentos. Para escoger cómo queremos el gráfico usaremos el argumento `type`. Los tipos posibles son los siguientes puntos (p), líneas (l), puntos y líneas (b), overplotted (o), como las barras de histograma (h), y pasos (s).

El título se puede crear añadiendo los argumentos de título dentro del paréntesis del comando del gráfico. Por ejemplo:

Figura 18

Scatterplots 2 a 2 de una matriz. Con un conjunto de scatterplots podemos ver representadas las relaciones 2 a 2 entre diferentes variables.

```
> # El titulo principal de un grafico de dispersion y de x
> # y las etiquetas de eje y
> plot(x, y, main = "Grafico de dispersion", xlab= "X variable", ylab= "variable Y")
```

Histograma

Los vectores pueden ser visualizados como histogramas. El comando para generar histogramas es `hist()`. Por defecto podemos disponer de 10 barras para la creación de nuestro gráfico, pero podemos cambiar el número de barras con el argumento `breaks()`. Por ejemplo, el código:

```
> hist (x)
> hist (x, breaks = 40)
```

produce las siguientes dos imágenes:

Figura 21. Histograma

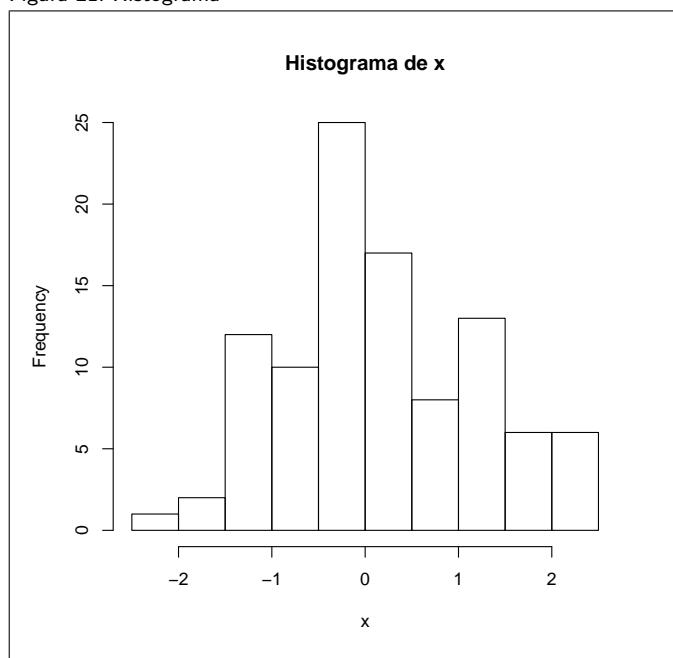


Figura 21

Histograma. Un Histograma es la representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados

Boxplot

Los Boxplots o Diagramas de caja se pueden producir usando el comando `boxplot()`. Este comando se puede usar para graficar un único vector como un diagrama de caja única o un data frame como varios diagramas en la misma figura. Si se representa una matriz con `boxplot()`, se tratará como un vector, y sólo se producirá un único diagrama de caja. Por ejemplo, el código:

```
> # Hacemos el grafico de un vector
> boxplot(x)
```

crea la imagen que sigue a continuación:

Figura 13. Boxplot

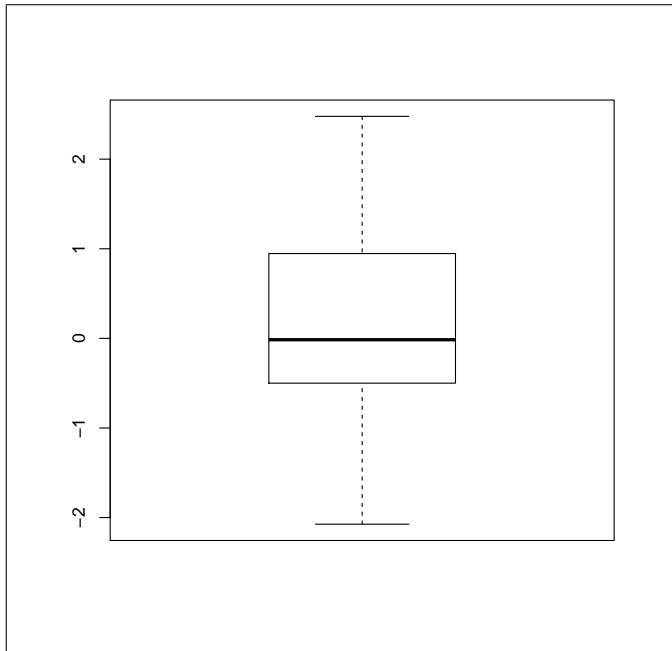


Figura 13

Boxplot. El boxplot es un diagrama con una “caja y bigotes” que representa el mínimo, el máximo, y los cuartiles del 25%, 50% y 75% de una variable.

Encontramos muchos argumentos diferentes para usar en un diagrama de caja, uno de ellos es `notch`. Podemos destacar la mediana de una distribución usando una marca producida por este argumento. Si las marcas de dos diagramas de caja no se superponen, esto puede destacar la gran diferencia que hay entre dos medias.

Scatter Plot

El diagrama de dispersión o Scatter plot se produce usando dos vectores en el comando `plot()` como ya se ha descrito más arriba.

Panel plots

Los gráficos de panel son figuras en las que se visualizan diferentes gráficos en un mismo marco. Por lo general todas las paneles son del mismo tipo, pero no necesariamente. Por ejemplo, el comando `pairs()` genera un panel de diagramas de dispersión para un conjunto de datos. Para definirlo se utiliza la instrucción `mffow` que permite determinar en cuántas filas y columnas se dispondrán los gráficos en el panel. Después de establecer las dimensiones, se genera un número igual de graficos. Estos se representan en el panel desde la parte superior izquierda a la esquina inferior derecha del panel. Por ejemplo, la figura siguiente se produjo

utilizando los siguientes instrucciones:

```
> # Las dimensiones del panel:  
> # 2 filas y dos columnas  
> par(mfrow = c(2,2))  
> hist(x, main = "A")  
> hist(x, rompe = 20, main = "B")  
> boxplot(x, main="C")  
> boxplot(dat, notch=T, main="D")  
> par(mfrow=c(1,1))
```

Figura 14. Gráfico de panel con cuatro figuras distintas

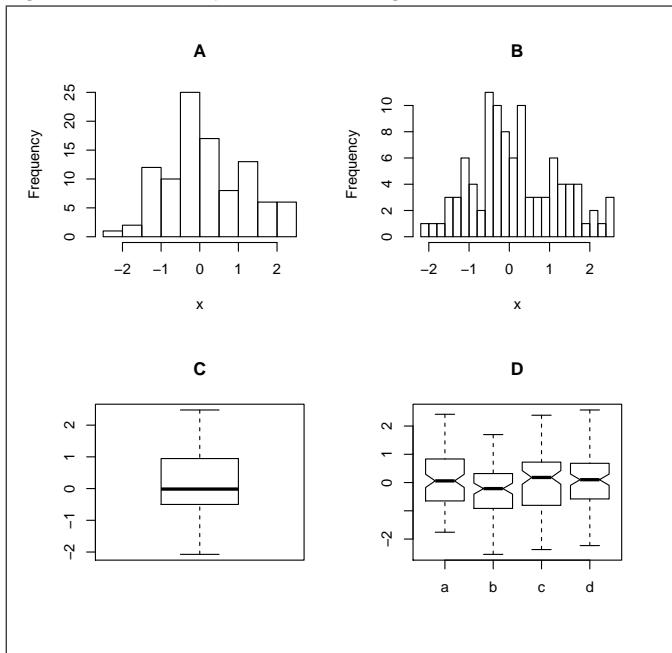


Figura 14

Un gráfico de panel permite visualizar simultáneamente varios gráficos del mismo o de distintos tipos.

Otras opciones gráficas

Como ya hemos comentado, la configuración gráfica se puede cambiar con el comando `par()`. A continuación, explicaremos algunas de las opciones más comúnmente utilizadas. Si los ajustes se aplican utilizando el comando `par()`, se aplicarán a todos los gráficos que hemos producido en la misma sesión de R. Si las opciones se aplican con `plotwise`, la configuración se aplica sólo al gráfico actual. Una de las tareas más comunes es cambiar el rango de los ejes. En R este trabajo se realiza mediante los argumentos `xlim` y `ylim`. Ambos argumentos toman un vector que tiene dos valores, el extremo inferior y el extremo superior del rango del eje. Por ejemplo,

```
> # Intervalo determinado por el rango de los datos  
> plot(x, y)  
> # Definimos el rango del eje  
> plot(x, y, xlim=c(-1, 1), ylim=c(-1, 1))
```

Otra tarea común es cambiar el símbolo y el tamaño de la fuente. Esto se logra con **cex-arguments**. **cex-arguments** cambia el tamaño del símbolo. Los argumentos **cex.axis**, **cex.label** y **cex.main** cambian el tamaño de la anotación del eje x e y, las etiquetas de eje y el título principal.

```
> plot(x, y, cex = 0.5, cex.axis = 0.5, cex.lab = 0.5)
```

Cambiar el tipo de gráfico utilizando el comando **plot()** ya se ha explicado más arriba. Si el tipo de gráfico es de líneas (**type = "l"**), el tipo y ancho de la línea se puede cambiar con los argumentos **lty** y **lwd**, respectivamente. Por ejemplo:

```
> # Generaci\'on de un grafico de l\'ineas discontinuas
> # con un grosor de 2
> z=1:length(x)
> plot (z, x, type = "l", lty = 2, lwd = 2) -2
```

La forma habitual para configurar el tipo de línea son los números 0-6 ((0 = blanco, 1 = sólido, 2 = discontinua, 3 = puntos, 4= puntos y guiones, 5 = guión largo, 6 = doble raya). El ancho de la línea del gráfico puede ser cualquier número positivo, aunque sea un número decimal (aunque no todos los tipos de gráficos soportan anchos de línea inferior a 1).

Incluir objetos en los gráficos

Se pueden añadir determinados objetos a la imagen de un gráfico ya creado. El comando **abline()** añade una línea horizontal o vertical en una posición determinada del gráfico ya existente:

```
> plot (x, y)
> abline (h = 0)
> abline (v = 0)
```

Los argumentos **h** y **v** especifican la posición horizontal o vertical de la línea, aunque puede añadirse cualquier linea, por ejemplo la recta de regresión entre **x** e **y**.

```
> plot (x, y)
> abline(lm(y~x))
```

Etiquetar los puntos de un gráfico

A veces es interesante conocer las etiquetas de las observaciones de un gráfico de dispersión. Las etiquetas se pueden agregar utilizando el comando `text()`. Se requiere tres argumentos para este comando, la coordenada x del texto, la coordenada y del texto y el texto a representar:

```
> plot (x, y)
> text (x, y-0.1, z, cex = 0.5)
```

Si hay necesidad de añadir nuevas observaciones al gráfico podemos usar el comando `points()`. Este comando tiene dos argumentos, `x` e `y` que serán las coordenadas de la nueva observación. El color (`s`) de la nueva observación (`s`) se especifica mediante el parámetro `col` y el tamaño y el tipo de símbolo se puede cambiar al mismo tiempo con los argumentos `cex` y `pch`:

```
> plot(x, y)
> points(-2, -2, col="red", cex=1.5, pch=19)
```

Las leyendas siempre se añaden después de realizar el gráfico. Para incluir una leyenda usaremos el comando `legend()`. Los argumentos que se necesita variar con la posición de la leyenda (`x`) y el texto que se escribe en la leyenda.

```
> # Grafico de dispersion con leyenda
> plot (x, y)
> points(1, -2, col = "red", pch = 19)
> legend ("bottomright", legend=c("Original", "Added later"), fill = c ("black", "red"))
```

Almacenamiento de imágenes

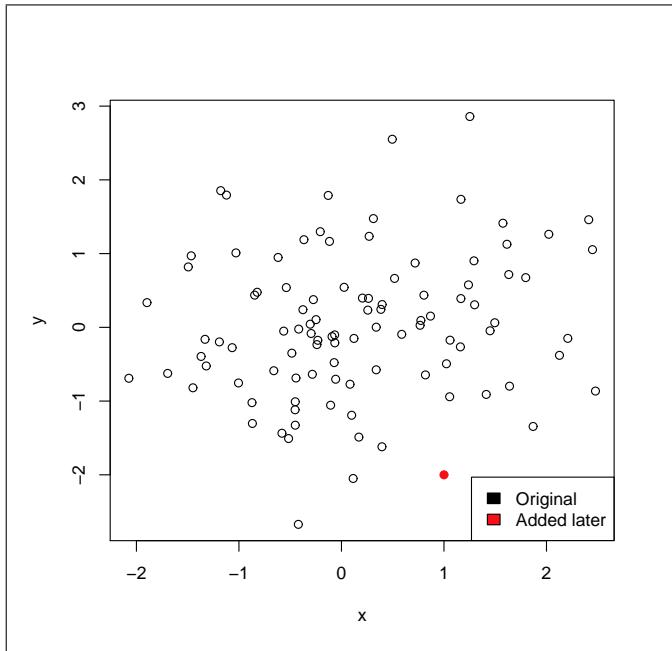
Existen distintos modos de guardar las imágenes en R. En Windows es posible activar las ventanas de la imagen (que se activa justo después de crear el gráfico), y guardarla utilizando el menú `File->Save as`. Podremos guardar la imagen en formato de metarchivo de Windows (no disponible en otros sistemas), PostScript, PDF, PNG, mapa de bits o JPG.

Figura 15

Diagrama de dispersión con puntos personalizados y leyenda.

De forma general, las imágenes también se pueden guardar en un determinado formato usando una instrucción antes de generar el gráfico. El dispositivo de gráficos se activa, el gráfico se genera y el dispositivo se cierra. Los dispositivos gráficos se abren con los instrucciones `PostScript()`, `pdf()`, `png()`, `bmp()`, `jpeg()`. El gráfico se genera como de costumbre, y los dispositivos se cierra con el comando

Figura 15. Personalización de un gráfico



`dev.off()`. Todos los instrucciones del dispositivo gráfico necesitan opciones ligeramente diferentes, así que es recomendable consultar la página de ayuda. Un ejemplo es:

```
> jpeg("scatterplot.jpg")
> plot(x, y)
> dev.off()
```

El gráfico que se genera con el comando `plot(x, y)`, se guarda en un archivo `scatterplot.jpg`.

2.2.9 Operaciones con matrices de datos

A menudo es importantes calcular la media o las desviaciones estándar de filas o columnas para extraer conclusiones relevantes en estudios bilógicos. Este tipo de cálculos en un matriz de datos se pueden hacer mediante bucles. Sin embargo, es mucho más conveniente usar el comando `apply()` que está orientado a aplicar de forma rápida una función a todas las filas o columnas de una matriz o data frame. Para ello se especifica el nombre de la matriz (o data frame), un valor que indica si se debe aplicar a las filas o a las columnas (1 para las filas y 2 para columnas), y por último el nombre de la función. Para ilustrar esto, calculamos la media de cada persona (la columna).

```
> gendat<-matrix(c(0.0125, 0.0625, 0.0250, 0.0125, 0.0625, 0.0650,
+ 0.0625, 0.0250, 0.0125, 0.0650, 0.0625, 0.0250),nrow=4,ncol=3,
+ dimnames=list(c("gene1", "gene2", "gene3", "gene4"),c("Ana",
```

```
+ "Teresa", "Elena")))
> apply(gendat, 2, mean)
```

```
Ana    Teresa    Elena
0.028125 0.053750 0.041250
```

Del mismo modo, la media de cada gen (fila) se puede calcular.

```
> apply(gendat, 1, mean)
```

```
gene1      gene2      gene3      gene4
0.02916667 0.06416667 0.05000000 0.02083333
```

Ocurre con frecuencia que se desea reordenar las filas de una matriz de acuerdo a un cierto criterio o, más específicamente, los valores de una determinada columna de un vector. Por ejemplo, para reordenar la matriz `gendat` de acuerdo con las medias de la fila, es conveniente guardarlos en un vector y utilizar el comando `order()`.

```
> meanexprsval <- apply(gendat, 1, mean)
> o<-order(meanexprsval, decreasing=TRUE)
> o
```

```
[1] 2 3 1 4
```

De este modo `gene2` aparece en primer lugar porque tiene la mayor media (0.06416667), luego viene `gene3` (0.05000000), seguido por `gene1` (0.02916667) y, finalmente, `gene4` (0.02083333). Ahora que hemos colocado los números del vector en orden, se puede reordenar toda la matriz mediante la especificación `o` en el índice de la fila.

```
> gendat[o, ]
```

```
Ana Teresa Elena
gene2 0.0625 0.0650 0.0650
gene3 0.0250 0.0625 0.0625
gene1 0.0125 0.0625 0.0125
gene4 0.0125 0.0250 0.0250
```

Para la selección de elementos de la matriz o data frame con una cierta propiedad tenemos varias opciones:

- Indicando en un vector los números de las filas o columnas a seleccionar

```
> gendat[c(1,2),]
```

```
Ana Teresa Elena  
gene1 0.0125 0.0625 0.0125  
gene2 0.0625 0.0650 0.0650
```

- Indicando en un vector los nombres de las filas o columnas a seleccionar

```
> gendat[c("gene1", "gene2"),]
```

```
Ana Teresa Elena  
gene1 0.0125 0.0625 0.0125  
gene2 0.0625 0.0650 0.0650
```

- Una tercera y más avanzada forma de hacerlo, es proceder a una evaluación en términos de TRUE o FALSE de los elementos lógicos de un vector. Por ejemplo, podemos evaluar si la media de la fila es positiva.

```
> meanexprsval > 0
```

```
gene1 gene2 gene3 gene4  
TRUE TRUE TRUE TRUE
```

Ahora podemos usar la evaluación de `meanexprsval > 0` en términos de los valores TRUE o FALSE como un índice de la fila.

```
> gendat[meanexprsval > 0,]
```

```
Ana Teresa Elena  
gene1 0.0125 0.0625 0.0125  
gene2 0.0625 0.0650 0.0650  
gene3 0.0250 0.0625 0.0625  
gene4 0.0125 0.0250 0.0250
```

Observe que esto selecciona los genes cuya evaluación es TRUE.

2.2.10 Utilización de scripts

Es aconsejable usar un editor de texto como el Notepad, Kate, Emacs o RStudio para crear un fichero (llamado script) con las instrucciones de R que se van a ejecutar, en líneas separadas (s). Este fichero debe tener extensión .r. Para

ejecutarlo se puede proceder de formas distintas. Una opción es abrirlo con un editor de texto estándar y copiar y pegar en la ventana de R las instrucciones a ejecutar. Otra posibilidad es abrir el fichero desde R (Menú:File->Open script) y ejecutarlo mediante el botón *Run line or selection*.

2.2.11 Más información

Se puede acceder a los manuales de R desde la misma web de ayuda (`help.start()`). En ésta, encontrará una introducción al programa R y una introducción a los instrucciones básicas estadísticos y de gráficos. En la página web <http://cran.r-project.org/other-docs.html> puede acceder a documentación diversa escrita por distintos usuarios de R.

Adicionalmente, se puede encontrar ayuda en la comunidad de usuarios de R en castellano, a través de su web y lista de correo-e:

Web: <http://demo.usar.org.es> (a partir de 2012: <http://r-es.org>)

Lista de correo-e: <https://stat.ethz.ch/mailman/listinfo/r-help-es>

2.3 El proyecto Bioconductor

Bioconductor es un proyecto de software libre (código y desarrollo abiertos), basado en el lenguaje de programación R, que proporciona herramientas para el análisis de datos genómicos de alto rendimiento.

El proyecto se inició en el año 2001 e incluye más de 25 desarrolladores entre Estados Unidos, Europa y Australia. En su versión 2.9, de Noviembre de 2011, incluye 517 paquetes, que presentan diferentes funcionalidades como análisis de microarrays, secuenciación y detección de SNPs, entre muchas otras.

2.3.1 Objetivos de Bioconductor

Los objetivos de Bioconductor son:

- Proporcionar acceso a potentes métodos estadísticos y gráficos para el análisis de datos genómicos.
- Facilitar la integración de metadatos biológicos (*GenBank*, *GO*, *LocusLink*, *PubMed*) en el análisis de datos experimentales.
- Permitir el desarrollo rápido de software extensible, interoperable y escalable.

- Promover la documentación de alta calidad y la investigación reproducible.
- Proporcionar formación en métodos de cálculo y estadística.

2.3.2 Paquetes

Para diseñar y distribuir el software de Bioconductor se usa el lenguaje R y su sistema de paquetes.

Como se ha visto anteriormente un paquete de R es una colección estructurada de código (R, C u otro), documentación, y/o datos para realizar tipos de análisis concretos. Según su contenido, los paquetes se pueden clasificar en:

- Paquetes de código (“software packages”), que proporcionan implementaciones especializadas de métodos estadísticos y gráficos.
- Paquetes de datos de diversos tipos como:
 - Metadatos biológicos (“annotation packages”), que contienen asignaciones (“mapping”) entre diferentes identificadores de genes (por ejemplo, “AffyID”, “GO”, “LocusID”, “PMID”), “CDF” e información de la sonda de secuencia de arrays de Affy. Ejemplos: `hgu95av2.db`, `GO.db`, `KEGG.db`.
 - Datos experimentales (“experiment packages”), que contienen código, datos y documentación para experimentos o proyectos específicos. Ejemplo: `yeastCC` (Spellman et al. 1988, yeast cell cycle), `golubEsets` (Golub et al. 2000, ALL/AML data).
- Paquetes de cursos, que contienen código, datos, documentación y ejercicios para el aprendizaje de un curso concreto. Ejemplos: `EMBO03`.

La figura 16 muestra algunos de los paquetes de Bioconductor agrupados por funcionalidades.

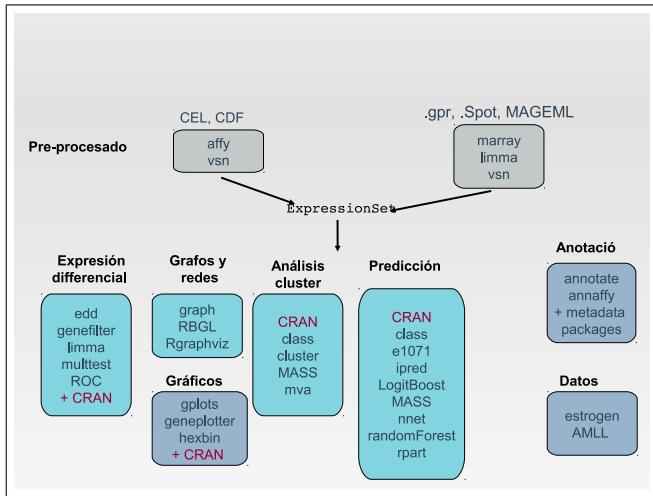
2.3.3 Programación orientada a objetos en R y Bioconductor

El proyecto Bioconductor adoptó el paradigma de la programación orientada a objetos (“OOP”) propuesto en 1998 por J.M. Chambers ([7]). Este diseño de clases y métodos orientados a objeto permite una representación eficiente y la manipulación de grandes y complejos conjuntos de datos biológicos de varios tipos. En el paquete `methods` de R se proporcionan herramientas para la programación mediante este mecanismo de clase/método.

Figura 16

Algunos de los paquetes de Bioconductor agrupados por funcionalidad

Figura 16. Algunos paquetes de Bioconductor



Para más información, ver www.omegahat.org/RSMMethods/index.html.

Clases y métodos

Una clase puede definirse como una representación (informática) abstracta de un tipo de objetos que pueden existir del mundo real. La clase refleja cómo imaginamos determinado objeto y qué información debe contener el mismo.

En Bioconductor una clase está formada por

- Unas componentes (“slots”) reservados para contener determinados datos.
- Unas funciones (“methods”) para manipularlos.

Las clases son representaciones abstractas que se concretan en *objetos*. Un objeto es una variable creada a partir de una clase, por lo que se le suele denominar *instancia* de la clase. La clase es la que determina la estructura y los valores iniciales que poseerán los objetos.

Un método es una función que realiza una acción sobre los datos (objetos). Cada método define como la acción particular debe comportarse dependiendo del tipo de argumentos que tenga. Los métodos permiten cálculos adaptables a cada tipo de datos en particular (es decir, a las clases). Hay funciones genéricas que actúan como un *distribuidor* (“dispatcher”), examinando los argumentos y determinando qué métodos son los más adecuados de invocar en cada caso.

Ejemplos de funciones genéricas en R son **plot**, **print**, **summary**.

El paquete de R **methods** contiene funciones tanto para la definición de nuevas classes y métodos (por ejemplo **setClass** y **setMethod**) como para el trabajo con

estos.

Entornos (“environments”) y clausuras (“closures”)

Un *entorno* es un objeto que contiene enlaces entre símbolos y valores. Es muy similar a una tabla *hash*. Los entornos son accesibles a través de las siguientes funciones:

- `ls(env=e)`, para obtener la lista de objetos del entorno e.
- `get('x', env=e)`, para obtener el valor del objeto x en el entorno e.
- `assign('x', y, env=e)`, para asignar a x el valor y dentro del entorno e.

Los entornos pueden asociarse a funciones. Cuando un entorno está asociado con una función, se utiliza para obtener valores de cualquiera de sus variables no vinculadas (“unbound variables”).

El término *clausura* se refiere a la unión (vinculación) de la función con el entorno que la *enjaula*. Paquetes de anotaciones, como `annotate`, `genefilter` y otros se basan en el uso de entornos y clausuras.

2.3.4 Dos clases importantes: `expressionSet` y `affyBatch`

Aunque la programación orientada a objetos no se ha impuesto completamente entre los usuarios de Bioconductor hay algunos conceptos que sí que se han generalizado y que es necesario –o como mínimo muy práctico– conocer para trabajar con Bioconductor. Se trata de la clase `expressionSet` y la clase `affyBatch`, directamente relacionada con ella.

La clase `expressionSet` se usa para representar datos procesados de arrays de dos colores o de un color. A su vez la clase está compuesta por objetos de otras clases como `assayData` (la matriz de expresiones, de *n* genes por *m* muestras), `phenoData` (las covariables de las muestras, una instancia de la clase `annotatedDataFrame`), `annotation` (nombre del paquete de anotaciones), `description` (información en formato MIAME) y `notes` (para comentarios adicionales).

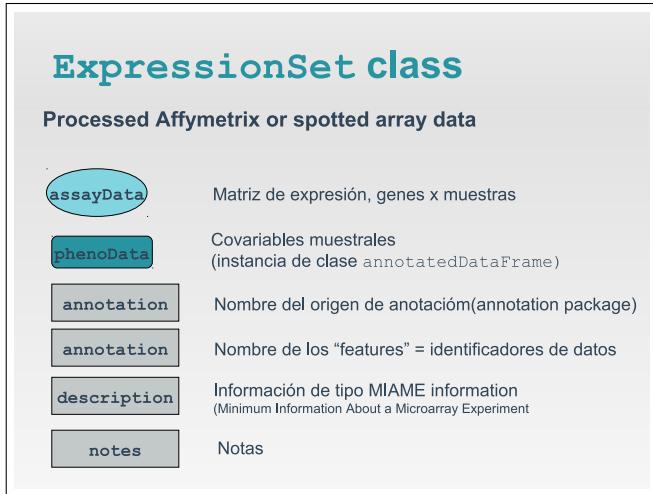
La figura 17 muestra la estructura de la clase `expressionSet`.

La clase `affyBatch` se usa para datos a nivel de intensidad de sonda para conjuntos de arrays (mismo CDF), y está integrada por objetos como `cdfName` (nombre del fichero CDF), `nrow` y `ncol` (dimensiones del array), `exprs` y `se.exprs` (matrices con las intensidades a nivel de sonda y sus errores estándar), `phenoData`,

Figura 17

Estructura interna de la clase `expressionSet`

Figura 17. La clase expressionSet



annotation, description y notes.

2.3.5 Primeros pasos: Instalación, cursos, vignettes

Instalación básica

La instalación de Bioconductor se ha descrito en el capítulo ?? por lo que no se repetirá aquí. Tan sólo recordar que se basa en dos instrucciones

```
> source("http://www.bioconductor.org/getBioC.R")
> getBioC()
```

En general, los paquetes de R se pueden instalar con la función `install.packages`. Estos paquetes solo necesitan ser instalados una vez, pero deben ser cargados en cada sesión nueva de R que se abra. Para cargar estos paquetes ya instalados, se usa el comando `library` de R.

```
> library(Biobase)
```

Para instalar cualquier paquete de Bioconductor, iniciar una sesión de R e introducir:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("nombre_del_paquete")
```

Para citar cualquier paquete desde R, teclear:

```
> citation("nombre_del_paquete")
```

Para actualizar los paquetes existentes, usar la función `update.packages`.

Cursos cortos

Bioconductor ofrece cursos rápidos en forma de elementos modulares de formación sobre software y metodología estadística, notas de conferencias, ejercicios prácticos de computación y otros paquetes de auto-formación disponibles online.

Las viñetas (vignettes)

La viñeta o “vignette” es el sistema de documentación adoptado por Bioconductor que se basa en la idea de *documentación reproducible*.

Consiste en un documento ejecutable que incluye una colección de bloques de código y bloques de comentarios de texto. Las viñetas ofrecen documentación estadística dinámica, integrada y reproducible que puede ser actualizada automáticamente si los datos o el análisis cambian.

Se generan usando la función `Sweave` de R.

Cada paquete de Bioconductor contiene al menos una viñeta, que aporta descripciones de sus funcionalidades. Las viñetas se encuentran en el subdirectorio `ddoc` de cada paquete instalado, y son accesibles desde el menú de ayuda. Pueden ser usadas de forma interactiva y están disponibles tambien de forma separada en el sitio web de Bioconductor (vease www.bioconductor.org).

Algunas funcionalidades relacionadas con las viñetas son `openvignette` de Biobase (un menú de viñetas disponibles con su visualización en PDF; `vExplorer` de `tkWidgets` (para uso interactivo de las viñetas), y `reposTools`.

Tanto los sitios web de R como de Bioconductor disponen de toda la documentación actualizada así como de listas de correo para resolver dudas de los usuarios.

2.3.6 Listado de paquetes usados o citados en el texto:

affy (“software package”) Métodos para arrays de oligonucleótidos de Affymetrix

El paquete contiene las funciones para el análisis exploratorio de arrays de oligonucleótidos. La dependencia de `tkWidgets` sólo se refiere a algunas funciones de conveniencia. El paquete `affy` es completamente funcional sin él.

affyPLM (“software package”) Métodos para el ajuste de modelos a nivel de sonda (“PLM”)

Un paquete que amplía y mejora la funcionalidad del paquete base **affy**. Tiene rutinas que hacen un uso intensivo de código compilado para más velocidad. Se centra en la aplicación de métodos para el ajuste de modelos a nivel de sonda y herramientas que usan estos modelos. Contiene herramientas de evaluación de la calidad basadas en PLM.

affyQCReport (“software package”) Generación de informes de control de calidad (“QC”) para objetos **affyBatch**

Este paquete crea informes de control de calidad (“QC”) para objetos **affyBatch**. El informe tiene por objeto permitir al usuario una rápida evaluación de calidad de un conjunto de matrices en un objeto **AffyBatch**.

annotate (“software package”) Anotación para microarrays

Uso de entornos R para gestionar las anotaciones.

arrayQualityMetrics (“software package”) Medidas de calidad en conjuntos de datos de microarrays.

Este paquete genera informes de medidas de calidad para contenedores de datos de microarrays de Bioconductor (**ExpressionSet**, **NChannelSet**, **AffyBatch**). El informe contiene tanto secciones generales como específicas de cada plataforma. Es compatible con plataformas de arrays de uno y dos colores.

Biobase (“software package”) Biobase: Funciones básicas de Bioconductor.

Funciones que son requeridas por muchos otros paquetes o que reemplazan ciertas funciones de R.

CMA (“software package”) Síntesis de clasificación basada en microarrays

Este paquete aporta una completa colección de algoritmos basados en microarrays para aprendizaje automático (“machine learning”) y estadística. Selección de variables, ajuste hiperparamétrico, evaluación y comparación, pueden ser ejecutados de forma combinada o paso a paso en un entorno fácil de usar.

cMAP (“annotation package”) Paquete de datos que contiene anotaciones para cMAP

Fichero de anotaciones para cMAP reunidas a partir de repositorios de datos públicos

e1071 (“CRAN package”) e1071: Funciones diversas del “Department of Statistics (e1071), TU Wien”

Funciones para el análisis de categorías latentes, transformada de Fourier de tiempo reducido, agrupamiento difuso (“fuzzy clustering”), máquinas de vectores de soporte (“support vector machines”), cálculo de la ruta más corta, agrupación en bolsas (“bagged clustering”), clasificador ‘naive’ de Bayes, ...

estrogen (“experiment package”)

Ejercicio de diseño factorial 2x2 para un curso rápido de Bioconductor

Datos de 8 genechips de Affymetrix, presentando un diseño factorial 2x2 (con 2 repeticiones por nivel)

gcrma (“software package”) Ajuste de ruido de fondo usando información de secuencia

Ajuste de ruido de fondo (“background correction”) usando información de secuencia

genefilter (“software package”) Métodos para el filtrado de genes de experimentos de microarrays

Algunas funciones básicas para el filtrado de genes

GO.db (“annotation package”) Conjunto de mapas de anotación que describen la “Gene Ontology”

Conjunto de mapas de anotación que describen la “Gene Ontology” al completo usando datos de GO

golubEsets (“experiment package”) exprSets para datos de leucemia de Golub

Representación de datos públicos de Golub con algunos datos de covarianza de procedencia desconocida hasta hoy por el responsable del paquete; ahora emplea el formato **ExpressionSet**

gplots (“CRAN package”) Herramientas varias de programación R para la representación gráfica de datos

Herramientas varias de programación R para la representación gráfica de datos

hgu95av2.db (“annotation package”) Datos de anotación de “Affymetrix Human Genome U95” (chip hgu95av2)

Datos de anotación de “Affymetrix Human Genome U95” (chip hgu95av2) procedentes de repositorios públicos.

KEGG.db (“annotation package”) Conjunto de mapas de anotación para “KEGG”

Conjunto de mapas de anotación para *KEGG* procedentes de la misma “Kyoto Encyclopedia of Genes and Genomes”

limma (“software package”) Modelos lineales para datos de microarrays

Análisis de datos, modelos lineales y expresión diferencial para datos de microarrays

multtest (“software package”) Remuestreo basado en pruebas de hipótesis múltiples

Procedimientos de pruebas de hipótesis múltiples basados en *Bootstrap* paramétrico y remuestreo por permutación (incluyendo métodos bayesianos empíricos) para el control de las tasas de error como “Familywise error rate (FWER)”, “generalized family-wise error rate (gFWER)”, “tail probability of the proportion of false positives (TPFP)”, y “false discovery rate (FDR)”.

PLIER (“software package”) Implementa el algoritmo PLIER de Affymetrix

El método PLIER (“Probe Logarithmic Error Intensity Estimate”) produce una señal mejorada teniendo en cuenta los patrones experimentales observados en el

comportamiento de las sondas y manejando de forma apropiada el error a nivel de los valores de señal extremos.

rpart (“CRAN package”) Particionamiento recursivo

Particionamiento recursivo y árboles de regresión

simpleaffy (“software package”) Análisis de alto nivel muy simple para datos de Affymetrix

Proporciona funciones de alto nivel para la lectura de archivos .CEL y datos fenotípicos de Affy, calculando entonces cosas simples, como “t-tests” y “fold changes”, con ellas. Hace un uso intensivo de la librería **affy**. También tiene algunas funciones de diagramas de dispersión y mecanismos para la generación de figuras de alta resolución para publicaciones.

tkWidgets (“software package”) Tk widgets basados en R

Widgets para proporcionar interfaces de usuario. Se requiere la instalación de **tcltk** para que funcionen.

yeastCC (“experiment package”) Datos de microarrays del ciclo celular de la levadura de Spellman et al. (1998) y Pramila/Breeden (2006)

ExpressionSet de los datos de experimentos del ciclo celular de la levadura, de Spellman et al. (1998)

ygs98.db (“annotation package”) Datos de anotaciones de “Affymetrix Yeast Genome S98” (chip ygs98)

Datos de anotaciones de “Affymetrix Yeast Genome S98 Array” (chip ygs98) obtenidos a partir de datos de repositorios públicos

ygs98cdf (“annotation package”) Paquete con el entorno del fichero **YG_-S98.cdf**.

Paquete con el entorno para la representación del fichero de **YG_S98.cdf** de “Affymetrix Yeast Genome S98”.

ygs98probe (“annotation package”) Datos de secuencias de sondas para los microarrays de tipo ygs98

Este paquete fue creado automáticamente por el paquete **AnnotationDbi** (versión 1.15.34). Los datos de secuencias de sondas se obtuvieron de www.affymetrix.com.

El fichero se llamó **YG-S98_probe.tab**.

3. Fundamentos de Estadística

3.1 Introducción

Cuando se realiza un estudio de investigación, el objetivo consiste en poder generalizar los resultados obtenidos en una muestra a una población (proceso conocido como inferencia). Para ello se estudia un reducido número de individuos (o más generalmente unidades experimentales), obtenidos de la población de forma independiente, con la idea de poder generalizar su comportamiento a la población de la cual esa muestra procede. Este proceso de inferencia se efectúa por medio de métodos estadísticos basados en la probabilidad. La población representa un conjunto (grande) de individuos que deseamos estudiar y generalmente suele ser inaccesible. Es, en definitiva, un colectivo homogéneo que reúne unas características determinadas. La muestra es el conjunto menor de individuos (subconjunto de la población accesible y limitado sobre el que realizamos las mediciones o el experimento con la idea de obtener conclusiones generalizables a la población). El individuo es cada uno de los componentes de la población y la muestra. La muestra debe ser representativa de la población lo que implica que cualquier individuo de la población en estudio debe haber tenido la misma probabilidad de ser elegido.

Las razones para estudiar muestras en lugar de poblaciones son diversas:

- Estudiar la totalidad de los pacientes o personas con una característica determinada en muchas ocasiones puede ser una tarea inaccesible o imposible de realizar.
- Aumentar la calidad del estudio. Al disponer de más tiempo y recursos, las observaciones y mediciones realizadas a un reducido número de individuos pueden ser más exactas y plurales que si las tuviésemos que realizar a una población.
- La selección de muestras específicas nos permitirá reducir la heterogeneidad de una población al indicar los criterios de inclusión y/o exclusión.

Lo que estudiaremos en cada individuo de la muestra son las variables (edad, sexo, peso, talla, tensión arterial sistólica, etcétera). Los datos son los valores que toma la variable en cada caso. Lo que vamos a realizar es medir, es decir, asignar valores a las variables incluidas en el estudio. Deberemos además concretar la escala de medida que aplicaremos a cada variable.

La naturaleza de las observaciones será de gran importancia a la hora de elegir el método estadístico más apropiado para abordar su análisis. Con este fin, clasificaremos las variables, a grandes rasgos, en dos tipos: cuantitativas y cualitativas.

- 1) Variables cuantitativas. Éstas pueden ser de dos tipos:
 - a) Variables cuantitativas continuas, si admiten tomar cualquier valor dentro de un rango numérico determinado (edad, peso, talla).
 - b) Variables cuantitativas discretas, si no admiten todos los valores intermedios en un rango. Suelen tomar solamente valores enteros (número de hijos, número de partos, número de hermanos, etc).
- 2) Variables cualitativas. Este tipo de variables representan una cualidad o atributo que clasifica a cada caso en una de varias categorías. La situación más sencilla es aquella en la que se clasifica cada caso en uno de dos grupos (hombre/mujer, enfermo/sano, fumador/no fumador). Son datos dicotómicos o binarios.

Como resulta obvio, en muchas ocasiones este tipo de clasificación no es suficiente y se requiere de un mayor número de categorías (color de los ojos, grupo sanguíneo, profesión). En el proceso de medición de estas variables, se pueden utilizar dos escalas:

 - a) Escalas nominales: ésta es una forma de observar o medir en la que los datos se ajustan por categorías que no mantienen una relación de orden entre sí (color de los ojos, sexo, profesión, presencia o ausencia de un factor de riesgo o enfermedad).
 - b) Escalas ordinales: en las escalas utilizadas, existe un cierto orden o jerarquía entre las categorías (grados de disnea, estadiaje de un tumor).

3.2 Análisis descriptivo

Una vez que se han recogido los valores que toman las variables de nuestro estudio (datos), procederemos al análisis descriptivo de los mismos.

3.2.1 Variables categóricas

Para variables categóricas, como el sexo, se quiere conocer el número de casos en cada una de las categorías, reflejando habitualmente el porcentaje que representan del total, y expresándolo en una tabla de frecuencias. Estas frecuencias se suelen expresar de forma gráfica mediante los diagramas de barra (barplots) y los gráficos circulares o de pastel (piecharts).

El ejemplo que usamos a continuación muestra cómo leer la secuencia de “X94991.1” de la especie *Homo sapiens* en el *GenBank*, para la construcción de una tabla de frecuencias y un gráfico de pastel. Un gen se compone de una secuencia de

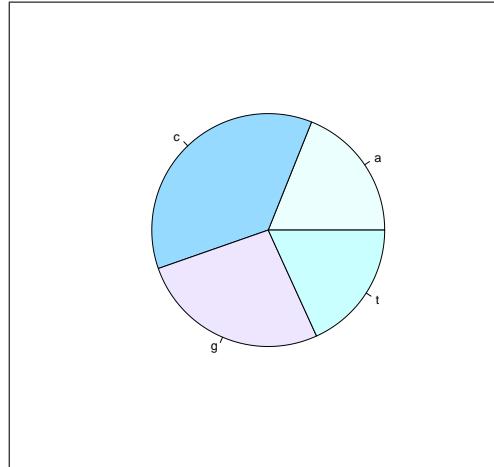
nucleótidos donde cada nucleótido puede ser de cuatro tipos: A, C, G, T. El número de veces que aparece cada nucleótido se pueden mostrar en una tabla de frecuencias. Por ejemplo, para el gen *Zyxin*.

```
> if(!(require("ape")))
+   install.packages(c("ape"),repo="http://cran.r-project.org",dep=TRUE)
> require(ape)
> print(t<- table(read.GenBank(c("X94991.1"),as.character=TRUE)) )
```

a	c	g	t
410	789	573	394

```
> pie(as.numeric(t))
```

Figura 18. Pie Chart o gráfico de pastel



Paquete APE en R

El paquete APE nos proporciona funciones para leer y manipular los árboles filogenéticos y las secuencias de ADN.

Figura 18

Pie Chart. El pie chart o gráfico de pastel es un gráfico de sectores que nos representa el porcentaje en que se dan los valores de una variable.

3.2.2 Variables numéricas:

Para variables numéricas, en las que puede haber un gran número de valores observados distintos, se ha de optar por un método de análisis diferente, respondiendo a las siguientes preguntas:

- 1) ¿Cómo se distribuyen los datos?
- 2) ¿Alrededor de qué valor se agrupan los datos?
- 3) Si suponemos que se agrupan alrededor de un número, ¿cómo lo hacen? ¿Muy concentrados? ¿Muy dispersos?

3.2.3 Gráficos:

Para desarrollar los ejemplos de este apartado usaremos los datos de expresión génica recogidos por Golub et al. (1999) [24]. Una serie de datos del conjunto Golub está contenido en el paquete `multtest`, que es parte de Bioconductor.

Referencia

Estos datos han sido procesados por los procedimientos descritos en Dudoit et al. (2002)

Los datos de este `dataset` consisten en valores de expresión génica (3051 filas de genes) de 38 pacientes enfermos de leucemia. A veintisiete pacientes se les diagnosticó leucemia linfoblástica aguda (ALL) y a once la leucemia mieloide aguda (AML). La clase de tumor viene dada por el vector numérico `golub.cl`, donde `ALL` se indica con el valor 0 y `AML` por el valor 1. Los nombres de los genes se recogen en la matriz `golub.gnames` donde sus columnas corresponden al número índice de los genes, el ID y el nombre, respectivamente.

Histogramas: Una primera aproximación de las variables se puede obtener a partir de la visualización de los datos. Para visualizar los datos con un histograma deberemos dividir el rango de valores en un número de intervalos iguales y representar las frecuencias por un intervalo que se dibujará como un rectángulo de altura proporcional a esta frecuencia. Como ejemplo vamos a crear un histograma a partir de los valores de expresión génica del gen "CCND3 Cyclin D3" de los pacientes con leucemia linfoblástica aguda, es decir aquellos que se etiquetan como "ALL", citados en el estudio de Golub et. al. (1999) ([24]). Para ello hemos de cargar los datos que se encuentran como ejemplo en la librería `multtest` mediante la instrucción `data(golub)`. Esta instrucción cargará en memoria la matriz de expresiones `golub`, junto a otros objetos entre los que se encuentra `golub.cl`, un vector de 0 y 1 correspondiente a los valores de "ALL" y "AML" respectivamente, para cada gen. Lo primero que se debe realizar es convertir este vector numérico a factor, para luego seleccionar el gen a estudiar (en nuestro caso corresponde a la fila 1042 de la matriz de expresiones) y poder realizar el histograma.

Paquete MULTTEST en Bioconductor

`multtest` contiene funciones para realizar múltiples test de hipótesis basadas en técnicas de remuestreo y además contiene los datos que vamos a trabajar

```
> if(!(require("multtest")))
+   {source("http://bioconductor.org/biocLite.R")
+   biocLite("multtest")
+ }
> require(multtest)
> data(golub)
> gol.fac<-factor(golub.cl, levels=0:1, labels=c("ALL", "AML"))

> hist(golub[1042, gol.fac=="ALL"])
```

Figura 19. Histograma

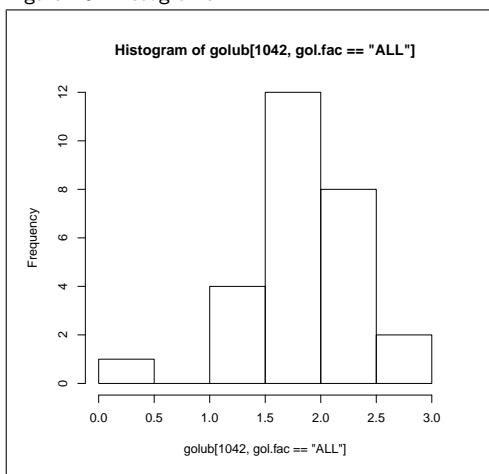


Figura 19

Histograma. Un Histograma es la representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados.

La función **hist()** en R divide nuestros datos en 5 intervalos de amplitud igual a 0.5. En el gráfico anterior podemos observar como los valores están más o menos simétricamente distribuidos alrededor de la media.

BoxPlot: El diagrama de caja se obtiene ordenando los n valores y calculando los siguientes cuartiles:

- Cuartil 25% (primer cuartil): valor del conjunto de datos para el que el 25% de los datos son menores que él.
- Cuartil 75% (tercer cuartil): valor del conjunto de datos para el que el 75% de los datos son menores que él.

Este gráfico consiste en un rectángulo (caja) que comprende los 2 cuartiles anteriores de cuyos lados (superior e inferior) se derivan dos segmentos respectivamente: uno hacia arriba y uno hacia abajo (llamados bigotes). Este rectángulo está dividido en 2 a partir del valor medio de los datos. Más allá de los bigotes o brazos de la caja se hallan las observaciones atípicas o extremas definidas como los valores que superan 1.5 veces el rango intercuartil.

Para visualizar este gráfico vamos a escoger un ejemplo bioinformático. Con la construcción de dos diagramas de caja para la expresión del gen **CCND3 Cyclin D3**, podemos tener una idea de la distribución de **ALL** y **AML**.

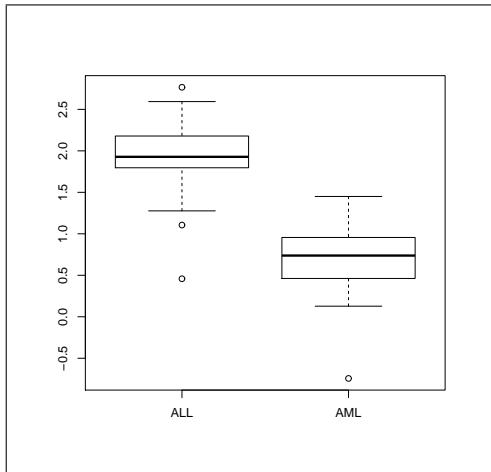
```
> boxplot(golub[1042,] ~ gol.fac)
```

Como podemos observar por la posición de las cajas los valores de expresión génica para **ALL** son mayores que los de **AML**. Además, dado que los dos recuadros alrededor de la mediana son más o menos igual de anchos, podemos decir que los datos están distribuidos razonablemente de forma simétrica alrededor de la mediana.

Figura 20

Boxplot. Un Boxplot o diagrama de caja es un gráfico en forma de caja que se crea a partir de los cuartiles y que nos permite detectar valores extremos y ver la simetría de una distribución.

Figura 20. Boxplot



3.2.4 Estadísticos descriptivos

Existen varias formas para describir la tendencia central, así como la dispersión de los datos. En particular, la tendencia central puede ser descrita por la *media* o la *mediana* y la dispersión por la *varianza*, la *desviación estándar* y el *rango intercuartílico* o la *desviación absoluta media*.

Medidas de tendencia central

Los estadísticos descriptivos de tendencia central más importantes son la media y la mediana. La media muestral de un conjunto de valores: $x_1 \dots x_n$ se define como:

$$\bar{x} = \frac{1}{n} \sum x_i = \frac{1}{n}(x_1 + \dots + x_n)$$

La mediana se define como el segundo cuartil y se denota $X_{0.5}$. Cuando los datos se distribuyen simétricamente alrededor de la media entonces la media y la mediana son iguales. Como los valores extremos no influyen en el tamaño de la mediana, es muy robusta frente a datos atípicos. La robustez es importante en bioinformática porque los datos están frecuentemente contaminados por valores extremos (outliers).

El término robustez se utiliza en estadística para hacer referencia a ciertas características deseables de los procesos estadísticos. Se dice que un proceso es robusto respecto de las desviaciones de los supuestos del modelo, cuando el proceso continúa funcionando bien, aún cuando, en mayor o menor extensión, los supuestos no se mantienen. En el ejemplo siguiente veremos como se calculan los cuartiles de una muestra. Para calcular los valores exactos de los cuartiles podemos usar el código siguiente con una secuencia de 0.00 a 1.00 con intervalos equivalentes a 0.25 cómo:

```
> pvec<-seq(0,1,0.25)
> quantile(golub[1042, gol.fac=="ALL"],pvec)
```

El primer cuartil es $X_{0.25} = 1.796$, el segundo $X_{0.50} = 1.928$ y el tercero $X_{0.75} = 2.179$.

Medidas de dispersión

Las medidas más importantes de dispersión son la desviación estándar, el rango intercuartílico y la desviación absoluta media. La desviación estándar es la raíz cuadrada de la varianza de la muestra, que se define como:

$$s^2 = \frac{1}{n-1} \sum (x_i - \bar{x})^2 = \frac{1}{n-1} [(x_1 - \bar{x})^2 + \dots + (x_n - \bar{x})^2]$$

Por lo tanto, la varianza es el promedio de la diferencia al cuadrado entre los valores de datos y la media de la muestra. La desviación estándar de la muestra es la raíz cuadrada de la varianza y puede interpretarse como la distancia de los datos a la media. La varianza y la desviación estándar no son estadísticos robustos frente a datos atípicos.

El rango intercuartílico se define como la diferencia entre el tercer y el primer cuartil, es decir, $R = x_{0.75} - x_{0.25}$.

3.3 Distribuciones de probabilidad importantes en estadística

Al iniciar el análisis estadístico de una serie de datos, y después de la etapa de detección y corrección de errores, el paso siguiente consiste en describir la distribución de las variables estudiadas y, en particular, de los datos numéricos. Además de las medidas descriptivas correspondientes, el comportamiento de estas variables puede explorarse gráficamente de un modo muy simple. Este comportamiento se puede estudiar, desde un punto de vista teórico, a partir del conocimiento de las principales distribuciones de probabilidad de las variables.

3.3.1 Distribuciones discretas

La distribución Binomial

La Distribución Binomial es fundamental y tiene muchas aplicaciones en medicina y bioinformática. La distribución binomial se ajusta a ensayos repetidos que generan una variable discreta dicotómica tales como éxito/fracaso, salud/enfermedad,

mutación/no mutación, purina/pirimidina, etc. Cuando se realizan n ensayos, el número de formas de obtener k éxitos viene dado por el coeficiente binomial:

$$\frac{n!}{k!(n-k)!} \cdot$$

La probabilidad binomial de k éxitos de n consiste en el producto de este coeficiente por la probabilidad de k éxitos y la probabilidad de $n-k$ fallos. Sea p la probabilidad de éxito en un único ensayo y x el valor (aleatorio) que denota el número de éxitos. Entonces la probabilidad P de obtener k éxitos en n ensayos ($X=k$) se expresará como:

$$P(X = k) = \frac{n!}{k!(n - k)!} p^k (1 - p)^{n - k}, \quad k = 0, \dots, n.$$

Al conjunto de estas probabilidades calculadas para cada posible valor de k se le llama función de densidad de probabilidad. El paquete R tiene la función `dbinom()` que proporciona el valor de estas probabilidades para cualquier valor de los parámetros n y p , por ejemplo para $x = 0$ y $x = 1$:

```
> n=100
> p=0.01
> dbinom(0,n,p)
```

[1] 0.3660323

```
> dbinom(1,n,p)
```

[1] 0.3697296

Podemos hacer un gráfico de la función de densidad probabilidad mediante:

```
> par(mfrow=c(1,1))
> plot(dbinom(0:100,n,p),xlab="Bits erroneos",type="h")
```

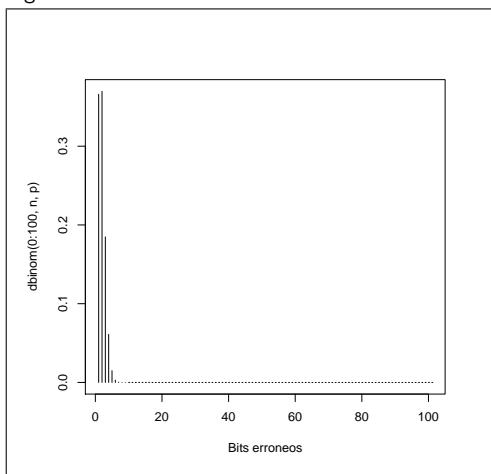
3.3.2 Distribuciones continuas

Las distribuciones continuas hacen referencia a aquellas variables que pueden tomar cualquier valor en un intervalo. Por lo tanto el cálculo de la distribución de probabilidades no puede hacerse de la misma forma que en el caso discreto, valor a valor. Es necesario conocer qué función matemática describe el comportamiento probabilístico de la misma. Esta función recibe el nombre de función de densidad y aunque su expresión no es importante en este curso, si se dará la forma de la misma para entender el comportamiento de la variable.

Figura 21

Función de densidad de probabilidad. La función de densidad se calcula a partir de las probabilidades de los valores de la variable.

Figura 21. Gráfico de la Función de densidad



La distribución Normal

Una de las distribuciones teóricas mejor estudiadas en los textos de bioestadística y más utilizada en la práctica es la distribución normal, también llamada distribución gaussiana. Su importancia se debe fundamentalmente a la frecuencia con la que distintas variables asociadas a fenómenos naturales y cotidianos siguen, aproximadamente, esta distribución. Carácter morfológicos (como la talla o el peso) o psicológicos (como el cociente intelectual) son ejemplos de variables de las que frecuentemente se asume que siguen una distribución normal. No obstante, y aunque algunos autores han señalado que el comportamiento de muchas variables en el campo de la salud puede ser descrito mediante una distribución normal, en la práctica puede resultar poco frecuente encontrar variables que se ajusten a este tipo de comportamiento.

El uso extendido de la distribución normal en las aplicaciones estadísticas puede explicarse, además, por otras razones. Muchos de los procedimientos estadísticos habitualmente utilizados asumen la normalidad de los datos observados. Aunque muchas de estas técnicas no son demasiado sensibles a desviaciones de la normal y, en general, esta hipótesis puede obviarse cuando se dispone de un número suficiente de datos, resulta recomendable contrastar siempre si se puede asumir o no una distribución normal. La simple exploración visual de los datos puede sugerir la forma de su distribución. No obstante, existen otras medidas, gráficos de normalidad y contrastes de hipótesis, que pueden ayudarnos a decidir de un modo más riguroso si la muestra de la que se dispone procede o no de una distribución normal. Cuando los datos no sean normales, podremos o bien transformarlos o emplear otros métodos estadísticos que no exijan este tipo de restricciones (los llamados métodos no paramétricos).

La distribución Normal viene caracterizada por la media y la varianza de la variable que se acostumbran a expresar con letras griegas: μ (**mu**) para la media y σ^2 (**sigma cuadrado**) para la varianza. La forma de indicar una distribución Normal es: $N(\mu; \sigma^2)$. Estos valores son desconocidos y más adelante se indicará como estimarlos. Esta distribución posee ciertas propiedades importantes que

conviene destacar:

- Tiene una única moda(valor más probable), que coincide con su media y su mediana.
- La función de densidad (curva normal o gaussiana) es asintótica al eje de abscisas. Por ello, cualquier valor de y es teóricamente posible. El área total bajo la curva es, por tanto, igual a 1.
- Es simétrica con respecto a su media . Según esto, para este tipo de variables existe una probabilidad de un 50% de observar un dato mayor que la media, y un 50% de observar un dato menor.
- La distancia entre la línea trazada en la media y el punto de inflexión de la curva es igual a una desviación típica (σ). Cuanto mayor sea σ , más aplanada será la campana.
- Existe un 95% de posibilidades de observar un valor comprendido en el intervalo $(\mu - 1.96 \cdot \sigma, \mu + 1.96 \cdot \sigma)$.
- La forma de la campana de Gauss depende de los parámetros μ y σ .
- La media indica la localización de la campana, de modo que para diferentes valores de μ la gráfica se desplazada a lo largo del eje horizontal. Por otra parte, la desviación estándar determina el grado de apuntamiento de la curva. Cuanto mayor sea el valor de σ , más se dispersarán los datos en torno a la media y la curva será más plana. Un valor pequeño de este parámetro indica, por tanto, una gran probabilidad de obtener datos cercanos al valor medio de la distribución.

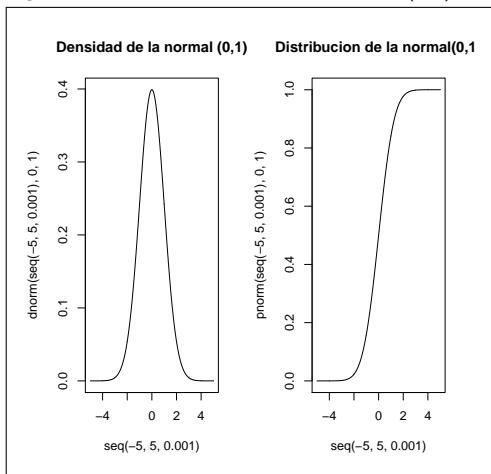
La función de densidad viene dada:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{[(x-\mu)^2/2\sigma^2]}, -\infty < x < \infty$$

La forma de calcular probabilidades es a partir de la *función de distribución* que se define $P(X \leq x)$, es decir la probabilidad, en la población, de obtener valores menores o iguales a x . Los gráficos de la función de densidad y distribución, son:

```
> plot(seq(-5,5,.001),dnorm(seq(-5,5,.001),0,1),type="l", main="Función de densidad ")
> plot(seq(-5,5,.001),pnorm(seq(-5,5,.001),0,1),type="l", main="Función de Distribución ")
```

Figura 22. Gráficos de distribución Normal(0,1)



De lo dicho anteriormente se determina que existe una distribución normal para cada valor de la media y la varianza. De entre todas, la que más se utiliza es la $N(0;1)$. Esta variable que se expresa mediante la letra Z, recibe el nombre de normal estandarizada o tipificada y se puede obtener a partir de cualquier normal mediante la transformación: $Z = (X - \mu)/\sigma$.

Figura 22

La distribución Normal. La distribución Normal, distribución gaussiana o de Gauss es una de las distribuciones continuas que aparece con más frecuencia en los fenómenos reales.

El Q-Q plot o gráfico de probabilidad normal El Q-Q (cuantil–cuantil) plot es un método que, entre otros procedimientos, sirve para visualizar la distribución de los valores de una variable. En este gráfico se enfrentan los cuantiles de los valores de la variable en la muestra contra los cuantiles correspondientes a la distribución normal. Se agrega una línea recta que representa los puntos que corresponden exactamente a los cuantiles de la distribución normal. Mediante la observación de la aproximación de los puntos a la línea, se puede evaluar en qué medida los datos se distribuyen normalmente. Es decir, cuanto más cerca de la línea, más probable es que los datos se distribuyan según una normal.

Un ejemplo de como producir un Q-Q plot de los valores de expresión génica del gen **CCND3 Cyclin D3** es usando el código siguiente:

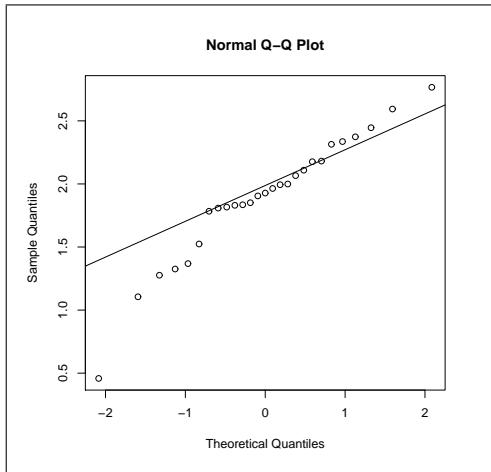
```
> qqnorm(golub[1042, gol.fac=="ALL"])
> qqline(golub[1042, gol.fac=="ALL"])
```

El ejemplo anterior ilustra un caso donde el grado de falta de normalidad es moderado por lo que no podemos sacar conclusiones.

Figura 23

Q-Q plot. El gráfico Q-Q plot es un método gráfico que muestra las diferencias entre la distribución de probabilidad de la que se ha extraído una muestra aleatoria y una distribución normal.

Figura 23. Q-Q Plot



Distribucion Chi-Cuadrado

La distribución Chi-cuadrado juega un papel importante en la verificación de hipótesis, tal como veremos en apartados posteriores. Desde un punto de vista teórico se puede definir de la siguiente manera: Si Z_1, \dots, Z_m son m variables normales tipificadas entonces la suma de cuadrados de las mismas:

$$x_m^2 = Z_1^2 + \dots + Z_m^2 = \sum Z_i^2; i = 1, \dots, m$$

siguen la distribución Chi-cuadrado con m grados de libertad.

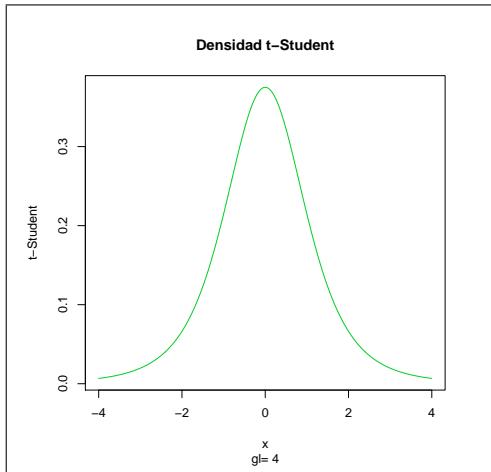
Distribución T

La distribución T (de Student) es una distribución útil en la verificación de hipótesis sobre las medias de las variables, en particular, cuando el tamaño muestral es menor que 30. Si los datos se distribuyen según una normal entonces los valores: $\frac{\sqrt{n}(\bar{x}-\mu)}{s}$ siguen una distribución T con $(n-1)$ grados de libertad. La distribución T se aproxima a una normal cuando el tamaño de la muestra es 30 o superior.

El gráfico que se genera a continuación es una T de Student que se asemeja a una normal:

```
> ngl<-4
> x<-seq(-4,4,length=5000)
> plot(x,dt(x,ngl), main=c("Densidad t-Student"),sub=paste("gl=",ngl), ylab="t-Student",type="l",col=3)
```

Figura 24. Gráfico T de Student



Distribución F

La distribución F de Fisher aparece de forma natural cuando se hace inferencia acerca de las varianzas de dos poblaciones. Más específicamente, si las varianzas de dos poblaciones son iguales ($\sigma_1^2 = \sigma_2^2$), se demuestra que el cociente de varianzas muestrales en dos muestras independientes de variables aleatorias distribuidas normalmente, $\frac{S_1^2}{S_2^2}$, sigue una distribución F con (n_1-1) y (n_2-1) grados de libertad. Donde s_1^2 es la varianza muestral del primer conjunto de n_1 observaciones y s_2^2 la varianza del segundo con n_2 observaciones.

Para finalizar esta sección haremos un resumen de las funciones de R usadas:

Tabla 3. Resumen de las funciones de R para el cálculo de la Densidad

Nombre de la Distribución	Parámetros	Densidad
Binomial	(n,p)	dbinom(x,n,p)
Normal	(μ, σ)	dnorm(x, μ, σ)
Chi-cuadrado	m	dchisq(x,m)
T de Student	m	dt(x,m)
F de Fisher	m	df(x,m,n)

Tabla 4. Resumen de las funciones de R para el cálculo de cuantiles y muestras de 10 valores

Nombre de la Distribución	Distribución	Cuantiles	Muestra Aleatoria
Binomial	pbinom(x,n,p)	qbinom(α, n, p)	rbinom(10,n,p)
Normal	pnorm(x, μ, σ)	qnorm(α, n, p)	rnorm(10,n,p)
Chi-cuadrado	pchisq(x,m)	qchisq(α, m)	rchisq(10,m)
T de Student	pt(x,m)	qt(α, m)	rt(10,m)
F de Fisher	pf(x,m,n)	qf(α, m, n)	rf(10,m,n)

Figura 24

T de Student. La T de student es una distribución relacionada con la inferencia acerca de la media a partir de los datos de una muestra.

3.4 Inferencia estadística

En los capítulos anteriores se utilizaron muchos parámetros poblacionales para definir familias de distribuciones teóricas. En cualquier entorno (empírico) de investigación los valores específicos de estos parámetros son desconocidos, por lo que deben estimarse. Una vez que se dispone de las estimaciones es posible probar estadísticamente hipótesis biológicamente importantes. Existen diferentes métodos para obtener las estimaciones de los parámetros a partir de las muestras. La mayoría se basan en la suposición de que las muestras provienen de poblaciones distribuidas normalmente. De entre los métodos, uno de los más utilizados es el llamado *Método de la máxima verosimilitud* que se basa en obtener la estimación de los parámetros de forma que se maximiza una función relacionada con la probabilidad de haber obtenido esa muestra concreta. Todos los métodos buscan estimaciones de los parámetros de forma que el estimador cumpla una serie de propiedades que se califican como “buenas”, para así garantizar que el valor obtenido se acerca de la mejor forma posible al verdadero valor. La estimación de los parámetros que se han visto hasta ahora, a partir de una muestra son:

- μ : Su estimador es la media muestral \bar{x}
- σ^2 : Su estimador es la varianza muestral s^2
- p : Su estimador es la proporción muestral o frecuencia relativa (f/n , siendo f el número de veces que se presenta la categoría en la muestra de tamaño n).

3.4.1 Contrastes de hipótesis

Un test estadístico es un procedimiento que permite, a partir de una muestra aleatoria representativa, extraer conclusiones que permitan aceptar o rechazar una hipótesis previamente establecida sobre el valor de un parámetro desconocido de una población. La hipótesis establecida se designa por H_0 y se llama hipótesis nula. Esta hipótesis se suele expresar indicando un valor concreto para algún parámetro poblacional ($H_0: \mu=0$). La hipótesis contraria se designa por H_1 y se llama hipótesis alternativa. Esta hipótesis se basa en la negación de la hipótesis nula y abarca distintas situaciones que dan nombre al tipo de contraste:

- Bilateral: $H_1: \mu \neq \mu_0$
- Unilateral: $H_1: \mu < \mu_0$ o bien $H_1: \mu > \mu_0$

Cada contraste sobre un parámetro lleva asociado un **estadístico de test** que es una variable aleatoria calculada a partir de la muestra ($T(X_1, \dots, X_n)$), asociada al test concreto, que describe la distribución de probabilidad del estimador del parámetro. Los pasos a seguir en la resolución de un contraste de hipótesis son los siguientes:

- 1) Plantear el contraste de hipótesis, definiendo la hipótesis nula y la hipótesis alternativa.
- 2) Especificar la probabilidad de error que se desea cometer cuando se rechaza la hipótesis nula siendo cierta (se denomina α o nivel de significación). Usualmente se escoge el valor 0.05 como nivel de significación más común.
- 3) Definir una medida de discrepancia entre la información que proporciona la muestra y la hipótesis H_0 . Esta medida de discrepancia se denomina estadístico del contraste y será una función de los datos muestrales y de la información de la hipótesis nula. Es estadístico del contraste debe seguir una distribución conocida bajo H_0 , que cumpla:
 - los valores altos sean poco probables cuando H_0 es cierta.
 - los valores pequeños sean altamente probables cuando H_0 es cierta.
- 4) Decidir que valores de este estadístico se consideran muy grandes, cuando H_0 es cierta, para que sean atribuibles al azar y por lo tanto se consideran inadmisibles cuando H_0 es correcta. El valor límite que marca la diferencia entre valores grandes y pequeños recibe el nombre de **valor crítico** y su valor queda determinado a partir del nivel de significación.
- 5) Estimar el parámetro a partir de la muestra y, a partir de él, calcular el valor del estadístico del test ($T_{exp}(X_1, \dots, X_n)$).
 - Si su valor es pequeño (pertenece a la región de aceptación), entonces se acepta la hipótesis H_0 .
 - Si es grande (pertenece a la región de rechazo), entonces se rechaza la hipótesis H_0 .

Tipos de Error en un contraste de hipótesis Al realizar un contraste se puede cometer uno de los dos errores siguientes:

- **Error tipo I**, el que se comete cuando se rechaza la hipótesis nula siendo cierta (*falso positivo*).
- **Error tipo II**, el que se comete cuando se acepta la hipótesis nula siendo falsa (*falso negativo*).

Debe tenerse en cuenta que sólo se puede cometer uno de los dos tipos de error y, en la mayoría de las situaciones, se desea controlar la probabilidad de cometer un error de tipo I. El nivel de significación es precisamente la probabilidad de cometer ese error de tipo I.

Una forma alternativa de tomar la decisión entre ambas hipótesis consiste en calcular el **p-valor**. El p-valor de un test para contrastar una hipótesis H_0 (aquí, que el gen no está diferencialmente expresado) se define como la probabilidad de

obtener un valor del estadístico del test tanto o más extremo que el valor que se ha obtenido sobre la muestra, suponiendo cierta la hipótesis nula, es decir

$$P[T(X_1, \dots, X_n) \geq T_{exp}(X_1, \dots, X_n) | H_0] = p^*$$

El p–valor nos informa sobre cuál sería el nivel de significación α más pequeño que nos hubiera permitido rechazar la hipótesis nula. La decisión consistirá, entonces, en rechazar la hipótesis nula si el p–valor es menor o igual al nivel de significación adoptado por el experimentador.

Sin querer profundizar en conceptos teóricos, este enfoque contiene posibles puntos problemáticos que es conveniente conocer para evitar cometer errores por mal uso o abuso de los conceptos. Concretamente:

- 1) Un p–valor bajo conlleva a rechazar H_0 , cuando puede ser que sea cierta por azar. Diremos en este caso que hemos declarado un falso positivo.
- 2) Los p–valores no son siempre correctos, dado que su validez depende de que se verifiquen ciertas suposiciones sobre los datos, como la normalidad. Cuando estas suposiciones fallan, los p–valores pueden ser completamente incorrectos.

El control de las probabilidades de error

Un test se suele organizar de forma que la probabilidad de obtener falsos positivos esté controlada, es decir que sea inferior al nivel de significación. Dicho control sin embargo no dice nada de la probabilidad de falsos negativos que puede ser muy alta sobretodo con pequeños tamaños muestrales. Es importante no perder de vista la tabla de decisión 3.4.1 para recordar que tipos de errores puede conllevar la selección de genes.

		DECISIÓN	
		Aceptar H_0	Rechazar H_0
REALIDAD	H_0	Decisión correcta	Error de TIPO I
	falsa	Error de TIPO II	Decisión correcta

Las probabilidades de cometer un error son

$$\underbrace{P(p^+ < \alpha | H_0 \text{ cierta})}_{P(\text{Falso positivo (FP)})}, \quad \underbrace{P(p^* > \alpha | H_0 \text{ falsa})}_{P(\text{Falso negativo (FN)})}.$$

Validez de los p–valores y p–valores válidos

Como se ha dicho, los p–valores dependen de que se verifiquen ciertas suposiciones como la independencia entre observaciones y normalidad de los datos. En general lo primero suele ser cierto –o asumible– mientras que lo segundo no tiene porque serlo y además es difícil de verificar en cualquier sentido (con 5 o 10 observaciones, lo que suele ser el caso de muchos estudios de microarrays no se puede hacer una prueba de normalidad de manera fiable.)

En estos casos se puede proceder de dos formas

- Mirar de comprobar gráficamente, de forma directa o indirecta, la hipótesis de normalidad.
- Recurrir a otros tipos de tests, como tests no paramétricos o tests de permutaciones que no precisan de la suposición de normalidad.

En general, dado que tanto los tests de permutaciones como los no paramétricos requieren de tamaños muestrales considerables, se suele utilizar distintas variantes del t–test, como las que se han discutido en la sección anterior.

3.4.2 Test t de una muestra

En casi todas las situaciones experimentales la desviación estándar de población, σ , es desconocida. En tales casos para realizar la prueba $H_0 : \mu = \mu_0$ frente $H_1 : \mu \neq \mu_0$ nos basaremos en calcular un estadístico de test con una distribución T :

$$t_{exp} = \frac{\sqrt{n}(\bar{x} - \mu_0)}{s}$$

Los pasos a seguir para decidir entre ambas hipótesis son:

- Determinar α .
- Calcular el estadístico del test a partir de la muestra.
- Calcular el p–valor asociado a este estadístico.
- Rechazar la hipótesis nula si el p–valor es menor que α . La probabilidad de error al tomar esta decisión será α .

3.4.3 Test t de dos muestras con varianzas distintas

Supongamos que se han observado los valores de una variable en dos grupos de pacientes (en dos condiciones experimentales) y que se desea verificar una hipótesis acerca de las medias poblacionales μ_1 y μ_2 .

En particular se desea verificar si: $H_0 : \mu_1 = \mu_2$ frente a $H_1 : \mu_1 \neq \mu_2$.

Este contraste también se puede formular como: $H_0 : \mu_1 - \mu_2 = 0$ frente a $H_1 : \mu_1 - \mu_2 \neq 0$.

Sean $x_1 \dots x_n$ y $y_1 \dots y_m$ las muestras, de de tamaño n y m respectivamente, de las dos poblaciones. El estadístico t del contraste, con distribución T es:

$$t_{exp} = \frac{(\bar{x} - \bar{y}) - (\mu_1 - \mu_2)}{\sqrt{s_1^2/n + s_2^2/m}} = \frac{(\bar{x} - \bar{y})}{\sqrt{s_1^2/n + s_2^2/m}}$$

El criterio de decisión con respecto a la hipótesis nula es idéntico a las pruebas anteriormente mencionadas. Se debe tener en cuenta que el valor de t es grande si la diferencia ($\bar{x} - \bar{y}$) es grande, las desviaciones estándar s_1 y s_2 son pequeñas y los tamaños muestrales son grandes. Esta prueba se conoce como el **test Welch** de dos muestras.

Como ejemplo seguiremos con los datos obtenidos de Golub et al. (1999) ([24]). Se sostiene que el gen CCND3 Cyclin D3 juega un papel importante con respecto a la discriminación de los pacientes con ALL sobre los que tienen AML. Intentaremos probar la hipótesis nula de medias iguales y varianzas distintas con el test t anterior:

```
> t.test(golub[1042,] ~ gol.fac, var.equal=FALSE)
```

```
Welch Two Sample t-test
```

```
data: golub[1042, ] by gol.fac
t = 6.3186, df = 16.118, p-value = 9.871e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
0.8363826 1.6802008
sample estimates:
mean in group ALL mean in group AML
1.8938826      0.6355909
```

el p–valor es 9.87e-06 con lo que rechazaremos la hipótesis de igualdad de medias.

3.4.4 Test t de dos muestras con varianzas iguales

Supongamos que estamos en la misma situación anterior pero ahora se sabe que las varianzas poblacionales (σ_1^2 y σ_2^2) son iguales. Para verificar el mismo test que en el apartado anterior el estadístico del test se basa en una estimación conjunta de la varianza a partir de las estimaciones en las dos muestras (*pooled sample variance*) que se calcula como:

$$S_p^2 = \frac{(n - 1)s_1^2 + (m - 1)s_2^2}{n + m - 2}$$

El valor del estadístico del test con distribución T se calcula en este caso como:

$$t_{exp} = \frac{(\bar{x} - \bar{y}) - (\mu_1 - \mu_2)}{S_p \sqrt{1/n + 1/m}} = \frac{(\bar{x} - \bar{y})}{S_p \sqrt{1/n + 1/m}}$$

Como ejemplo analizaremos el contraste de hipótesis anterior pero suponiendo varianzas iguales (`var.equal=TRUE`):

```
> t.test(golub[1042,] ~ gol.fac, var.equal = TRUE)
```

Two Sample t-test

```
data: golub[1042, ] by gol.fac
t = 6.7983, df = 36, p-value = 6.046e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
0.8829143 1.6336690
sample estimates:
mean in group ALL mean in group AML
1.8938826      0.6355909
```

El p-valor es de 6.046e-08, la conclusión es rechazar la hipótesis nula de medias poblacionales iguales. En este caso el p-valor es ligeramente menor que el de la prueba anterior. En caso de duda sobre la validez de la hipótesis de igualdad de varianzas de la población, se puede comprobar utilizando el siguiente test.

3.4.5 Test F de igualdad de varianzas

El test anterior se basa en que las varianzas poblacionales son iguales. Al ser estas varianzas desconocidas se debe plantear un nuevo test de hipótesis que permita verificar esta suposición. Es decir, se desea verificar: $H_0 : \sigma_1^2 = \sigma_2^2$ frente

a $H_1 : \sigma_1^2 \neq \sigma_2^2$. Para contrastar estas hipótesis se debe partir del cálculo de las varianzas muestrales, y se debe calcular es estadístico f:

$$f_{exp} = \frac{s_1^2}{s_2^2}$$

que sigue una distribución F de Fisher con grados de libertad ($n-1, m-1$).

Si seguimos con el ejemplo anterior, en este caso la hipótesis nula para el gen CCND3 Cyclin D3 será que la varianza de los pacientes con ALL es igual a la de los pacientes con ALM y esto se puede probar con la función `var.test()`:

```
> var.test(golub[1042,] ~ gol.fac)
```

```
F test to compare two variances
```

```
data: golub[1042, ] by gol.fac
F = 0.7116, num df = 26, denom df = 10, p-value = 0.4652
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.2127735 1.8428387
sample estimates:
ratio of variances
0.7116441
```

En este caso con un p-valor de 0.4652 no podemos rechazar la hipótesis nula de igualdad de varianzas.

3.4.6 Test Binomial

Supongamos que estamos interesados en verificar la hipótesis de que la probabilidad de que se presente una característica en la población es un determinado valor p_0 , frente al hecho de que esta probabilidad sea mayor. El contraste de hipótesis asociado será: $H_0 : p = p_0$ frente a $H_1 : p > p_0$. Si el experimento indica que la característica se ha presentado k veces en una muestra de tamaño n y suponiendo una distribución binomial, se podrá verificar la hipótesis nula a partir del cálculo del p-valor $P(X \geq k)$. Si es menor que el nivel de significación prefijado (0.05 por ejemplo) se rechaza ésta.

En el caso en que n sea grande ($n \geq 30$) se suele recurrir a la propiedad de que la *distribución Binomial* se approxima a la *Normal* cuando n tiende a infinito.

Por lo tanto el estadístico del test

$$Z = \frac{[(k/n) - p_0]}{\sqrt{p_0(1 - p_0)}}$$

seguirá una distribución aproximadamente normal. En el caso de que n sea más pequeño que 30 se debe seguir un procedimiento distinto. Supongamos que estamos estudiando secuencias cortas de ARN (microARNs) y nos dicen que “se acepta” que un microARN de longitud 22-mer contiene unas 18 purinas, lo que nos sugiere probar la hipótesis nula $H_0 : p = 0.7$ en contra de $H_1 : p > 0.7$:

Para realizar este test podemos usar la función `binom.test()` de la siguiente forma:

```
> binom.test(18, 22, p = 0.7, alternative = c("greater"), conf.level = 0.95)
```

```
Exact binomial test

data: 18 and 22
number of successes = 18, number of trials = 22, p-value = 0.1645
alternative hypothesis: true probability of success is greater than 0.7
95 percent confidence interval:
0.6309089 1.0000000
sample estimates:
probability of success
0.8181818
```

El p-valor de 0.1645 es mayor que el nivel de significación 0.05, por lo que la hipótesis nula no puede ser rechazada.

3.4.7 Test Chi-cuadrado

A menudo se desea verificar una hipótesis sobre más de una probabilidad. Es decir $H_0 : (p_1 \dots p_m) = (p_{01}, \dots, p_{0m})$ frente a $H_1 : (p_1 \dots p_m) \neq (p_{01}, \dots, p_{0m})$ donde $p_1 \dots p_m$ son probabilidades poblacionales correspondientes a m categorías y p_{01}, \dots, p_{0m} son los valores teóricos que se les suponen. Para cada valor indicado en H_0 se puede calcular el número esperado de ocurrencias en n ensayos independientes: $e_i = np_i$

El estadístico para verificar el contraste se basa en la diferencia entre los valores observados (o_i) y los esperados (e_i) para cada categoría:

$$q = \sum \frac{(o_i - e_i)^2}{e_i}$$

este estadístico sigue una distribución *Chi-cuadrado* con $m-1$ grados de libertad. El p-valor se calcula como $P(\chi^2 > q)$.

El test Chi-cuadrado es aplicable si el tamaño muestral es relativamente grande o, lo que es similar, si los valores esperados son mayores que 5. En caso contrario se debe acudir a otros tests como el que se detalla a continuación.

Un test relacionado con el test Chi-cuadrado y que se utiliza con frecuencia en Bioinformática es el llamado *Test exacto de Fisher*. Este test permite analizar si dos variables dicotómicas están asociadas cuando la muestra a estudiar es demasiado pequeña y no se cumplen las condiciones necesarias para que la aplicación del test sea adecuada. Supongamos que tenemos dos categorías dicotómicas, A y B, y la siguiente tabla de frecuencias:

Tabla 5. Tabla de Frecuencias

		Categoría A
Categoría B	Presente	Ausente
Presente	f_{11}	f_{12}
Ausente	f_{21}	f_{22}

Tabla de Frecuencias

La Tabla de frecuencias es la recopilación en forma de tabla de la distribución de frecuencias de una variable

este test se basa en el cálculo del llamado *odds ratio* o *razón de probabilidad* $f_{11}f_{22}/(f_{12}f_{21})$. La hipótesis nula del *test exacto de Fisher* plantea que el *odds ratio* es igual a 1 y la hipótesis alternativa es que difiere de 1.

Veamos un ejemplo de este test (usado con frecuencia en Bioinformática) y del cálculo del “odds ratio”. Supongamos que las frecuencias de oncogenes para el Cromosoma 1 es igual a $f_{11} = 300$ y la de los que no lo son es de $f_{12} = 500$. Si consideramos el genoma $f_{21} = 3000$ y $f_{22} = 6000$. La hipótesis de la que el odds ratio es igual a uno la podemos probar de la siguiente manera:

```
> dat <-matrix(c(300,500,3000,6000),2,byrow=TRUE)
> fisher.test(dat)
```

Fisher's Exact Test for Count Data

```
data: dat
p-value = 0.01912
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
1.029519 1.396922
sample estimates:
odds ratio
1.19996
```

Puesto que el p-valor (0.01912) es menor que el nivel de significación, podemos decir que significativamente hay más oncogenes en el Cromosoma 1 comparado con el del genoma.

3.4.8 Test de Normalidad

Existen diversos procedimientos para probar la hipótesis de que un conjunto de datos se distribuye normalmente. La prueba de *Shapiro-Wilks* se basa en el grado de linealidad en un gráfico Q-Q (Lehmann, 1999, p.347).

Como ejemplo del uso de la prueba Shapiro-Wilks probaremos la hipótesis de que el valor de la expresión genética del los genes ALL del CCND3 Cyclin D3 se distribuye normalmente:

Referencia bibliográfica

E.L. Lehmann. Elements of Large-Sample Theory, Springer. 1999

```
> shapiro.test(golub[1042, gol.fac=="ALL"])

Shapiro-Wilk normality test

data: golub[1042, gol.fac == "ALL"]
W = 0.9466, p-value = 0.1774
```

3.4.9 Test de rangos de Wilcoxon

Hemos visto, en anteriores apartados, que la comparación de las medias de dos poblaciones normales se realizaba mediante el test t. Sin embargo si las poblaciones no se distribuyen siguiendo una distribución normal debido a una asimetría de la curva de la distribución el test t no es aplicable. En estos casos se utilizan tests basados en los rangos ya que no requieren suposiciones específicas sobre la distribución de la variable analizada.

Supongamos que deseamos comparar dos poblaciones de las que se han obtenido las muestras: (x_1, \dots, x_n) con distribución F_x y (y_1, \dots, y_n) con distribución F_y . Las dos distribuciones F_x y F_y son desconocidas. Si no hay diferencia en el comportamiento de la variable medida en ambas poblaciones (hipótesis nula), las dos distribuciones son idénticas. $H_0 : F_x = F_y$. La idea del *test de Wilcoxon-Mann Whitney* es la siguiente: si unimos las dos muestras y ponemos los valores en orden, la alternancia entre las X_i y las Y_j debería ser bastante regular. Tendríamos dudas sobre H_0 si los Y_j fueran en general más grandes que los X_i , o más pequeños, o más frecuentes en ciertos tramos de la sucesión de valores.

El estadístico U de *Wilcoxon-Man Whitney* se basa en asignar rangos (número

de orden en una única muestra obtenida uniendo las dos y ordenando de menor a mayor) a cada valor. Si la hipótesis nula es cierta, la suma de los rangos de las X_i y de las Y_j debería ser parecida. El estadístico de test para esta prueba compara estas sumas para decidir sobre la hipótesis.

Por ejemplo, si queremos realizar la comparación anterior de los grupos ALL y AML pero sin suponer normalidad el código será:

```
> wilcoxon.test(golub[1042,] ~ gol.fac)
```

3.5 Corrección para pruebas múltiples (*Multiple testing*)

Es importante tener en cuenta el efecto que tiene la realización de múltiples tests simultáneos en todos aquellos tests estadísticos que calculan un p-valor. Si en un test el p-valor es 0.05 entonces se tiene una probabilidad de 5% de dar un error de tipo I (falso positivo), por lo tanto, si realizamos 10000 pruebas a la vez se esperaran unos 500 ($=10000 \times 0.05$) errores de tipo I (es decir en 500 tests diremos que hay significación cuando en realidad no es así). Este resultado no puede considerarse aceptable. Una posibilidad es utilizar la *corrección de Bonferroni* para reducir el umbral para detectar significación a un nivel en el que haya sólo un 5% de probabilidad de cometer uno o más errores de tipo I entre todas las pruebas. Este nuevo punto de corte será $0.05/10000 = 5E10 - 6$. Este valor es muy estricto, ya que en muchas situaciones se puede admitir un número mayor de errores de tipo I.

Un cambio de filosofía consiste en determinar el número de errores de tipo I (falsos positivos) que se está dispuesto a aceptar al realizar m pruebas independientes. El *False Discovery Rate (FDR)* es un método estadístico utilizado cuando se presentan múltiples hipótesis. En una lista de hipótesis rechazadas el *FDR* controla la proporción esperada de hipótesis nulas incorrectamente rechazadas. El procedimiento para efectuar el control de los falsos positivos mediante el *FDR* es el siguiente:

Si consideramos $H_1 \dots H_m$ las hipótesis y $P_1 \dots P_m$ sus p-valores ordenados de manera creciente. Para un FDR (q) dado encontraremos el menor k a partir del cual:

$$P_k < \frac{k}{m} * q$$

donde k es el número de pruebas aceptadas hasta el momento, m es el número total de pruebas realizadas y q es la tasa deseada de FDR. Para $k > 1$, esta corrección es menos estricta que realizar la corrección de Bonferroni. Si no se encuentran tests significativos utilizando Bonferroni o la corrección por FDR, entonces se puede mirar a los que tienen el menor p-valor. Se puede obtener una

estimación de la *FDR* si se multiplica su p-valor por el número de tests múltiples en el experimento.

3.6 Análisis de la varianza

El propósito del análisis de la varianza (*ANOVA*) es comprobar si existen diferencias significativas entre las medias de varios grupos. Si se comparan solamente dos medias, el *ANOVA* proporciona el mismo resultado que el test t. Sin embargo, a diferencia de la test t, el *ANOVA* no permite saber cuáles de los grupos son significativamente diferentes entre sí, ya que solo detecta la existencia de estas diferencias. El nombre *ANOVA* se deriva del hecho de que a fin de verificar una comparación de medias este análisis se lleva a cabo a partir de la estimación y comparación de unas varianzas específicas. La base del procedimiento consiste en dividir o descomponer la variabilidad total de los datos en componentes de variabilidad debidas a cada uno de los factores que, en nuestro experimento, pueden afectar a la misma. Estas fuentes de variabilidad se cuantifican como sumas de cuadrados de desviaciones respecto a la media.

3.6.1 Análisis de la varianza de un factor

El caso más sencillo es el *modelo de un factor*, también conocido por “one way anova”, y corresponde a la comparación de k grupos de individuos que han recibido tratamientos distintos. El contraste de hipótesis es:

$$\begin{aligned} H_0 &: \mu_1 = \mu_2 = \dots = \mu_k \\ H_1 &: \mu_i \neq \mu_j \text{ para alguna } i,j \end{aligned}$$

En el modelo ANOVA de un factor se supone que cada observación Y_{ij} (observación j-ésima del grupo i) puede expresarse como:

$$Y_{ij} = \mu_i + \epsilon_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

A los valores ϵ_{ij} se les llama residuos, y son las desviaciones de cada observación a la media del grupo del que proviene. Se les supone una distribución normal de media 0. Los valores α_i se llaman efectos de cada nivel y representan una medida de la desviación de los datos del grupo i respecto a la media global.

La hipotésis nula y alternativa anteriores pueden expresarse como:

$$\begin{aligned} H_0 &: \alpha_1 = \alpha_2 = \dots = \alpha_k = 0 \\ H_1 &: \alpha_i \neq 0 \text{ para alguna } i \end{aligned}$$

Para contrastar H_0 , usaremos la noción de variabilidad, que esencialmente coincide con la de dispersión. Para medir la variabilidad de los datos Y_{ij} utilizaremos las *suma de cuadrados totales (SCT)*, y la descompondremos en suma de dos términos: la *suma de cuadrados residual o intra-grupos (SCR)*, que tiene que ver con la variabilidad dentro de cada nivel de factor, y la *suma de cuadrados explicada o entre-grupos (SCE)*, que mide la variabilidad debida a las diferencias entre la media de cada factor y la media global. Más concretamente, se cumple:

$$SCT = SCR + SCE$$

o sea

$$\sum(Y_{ij} - \bar{Y})^2 = \sum(Y_{ij} - \bar{Y}_i)^2 + \sum(\bar{Y}_i - \bar{Y})^2$$

siendo \bar{Y} la media de todos los datos y \bar{Y}_i la media del grupo i .

Claramente, si H_0 es cierta, entonces SCE será pequeña frente a SCT. De hecho se llama porcentaje de variabilidad explicada a $(SCE/SCT) \times 100$.

Cada una de las sumas de cuadrados anteriores permite una estimación de la varianza o *Cuadrado medio*:

- Cuadrado Medio Entre grupos: $CM_E = \frac{SCE}{k-1}$
- Cuadrado Medio Residual: $CM_R = \frac{SCR}{N-k}$, siendo N el número total de datos.

el estadístico de test se basa en la comparación de estas dos estimaciones de la varianza y, por lo tanto, seguirá una distribución F:

$$F = \frac{CM_E}{CM_R}$$

En general podemos decir que H_0 será aceptada si el valor del estadístico se encuentra alrededor del 1. Si es suficientemente mayor que 1, entenderemos que el factor que hemos introducido está realmente explicando las diferencias que observamos entre los valores de la variable Y, y por tanto que efectivamente hay cierta relación entre Y y el factor que determina los grupos, con lo cuál H_0 será falsa. Observemos también que si se rechaza H_0 , ello no implica que todas las medias poblacionales sean distintas entre sí, sino simplemente que alguna(s) de ellas es diferente a las demás. De hecho, pueden localizarse los diferentes grupos que aparecen entre los niveles del factor a partir de la realización de comparaciones dos a dos usando un método de comparaciones múltiples.

El ANOVA requiere el cumplimiento de los siguientes supuestos:

- Las distribuciones de probabilidad de la variable Y en cada grupo son normales.
- Las k muestras sobre las que se aplican los tratamientos son independientes.
- Las poblaciones tienen todas igual varianza (homoscedasticidad).

Como ejemplo utilizaremos los datos **ALL** del paquete **ALL** y, específicamente los valores de expresión del oncogen *1866_g_at* en las células B. Si se realiza un gráfico de los datos se observa que los niveles de expresión difieren en los tres estadios de la enfermedad, por lo tanto estudiaremos si existen realmente diferencias entre los tres grupos.

```
> if (!require("ALL"))
+   {source("http://bioconductor.org/biocLite.R")
+     biocLite("ALL")
+   }
> require(ALL)
> data(ALL)
> ALLB123<-ALL[, ALL$BT %in% c("B1", "B2", "B3")]
> y<-exprs(ALLB123)[["1866_g_at", ]]
> anova(lm(y~ALLB123$BT))
```

Analysis of Variance Table

```
Response: y
          Df  Sum Sq Mean Sq F value    Pr(>F)
ALLB123$BT  2  5.4563  2.72813 19.848 1.207e-07 ***
Residuals  75 10.3091  0.13745
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> summary(lm(y~ALLB123$BT))
```

Call:

```
lm(formula = y ~ ALLB123$BT)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.59070	-0.25251	-0.07008	0.16395	1.29635

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	4.58222	0.08506	53.873	< 2e-16 ***

```

ALLB123$BTB2 -0.43689   0.10513  -4.156 8.52e-05 ***
ALLB123$BTB3 -0.72193   0.11494  -6.281 2.00e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3707 on 75 degrees of freedom
Multiple R-squared: 0.3461,          Adjusted R-squared: 0.3287
F-statistic: 19.85 on 2 and 75 DF,  p-value: 1.207e-07

```

El p-valor del test F es 1.207e-07 por lo tanto se rechaza la hipótesis nula de igualdad de medias. Los coeficientes del modelo representan: (Intercept) es el efecto de B1, ALLB123\$BTB2 es la diferencia entre B2 y B1 y ALLB123\$BTB3 es la diferencia entre B3 y B1. De los t-tests individuales para estos coeficientes se concluye que la media de B1 es significativamente distinta de cero y que las diferencias entre B2 y B1 y B3 y B1 también lo son.

3.6.2 ANOVA de más de un factor

En análisis de la varianza se puede extender para permitir al investigador planificar un trabajo para evaluar el efecto combinado de dos o más variables de forma simultánea en el resultado medido, obteniéndose también información en cuanto a la posible interacción entre los diversos factores. El término "experimento factorial" o "arreglo factorial" se refiere a cómo se asignan los tratamientos que se quieren comparar. Para ello es necesaria una selección previa de los factores a estudiar, sus niveles y la combinación de ellos. La necesidad de estudiar conjuntamente varios factores obedece a la posibilidad de que el efecto de un factor cambie según los niveles de otros factores, esto es, que los factores interactúen, o exista interacción.

Si Y_{ijk} es la respuesta para la k -ésima unidad experimental del nivel i de A y j de B, entonces el modelo asociado a un experimento con dos factores: A con a niveles y B con b niveles, asignados completamente al azar es:

$$Y_{ijk} = \mu + \tau_i + \beta_j + \gamma_{ij} + \epsilon_{ijk}$$

En este modelo τ_i es el efecto del nivel i de A, β_j es el efecto del nivel j del factor B y γ_{ij} es el efecto de la interacción. Conocer la interacción es más útil que conocer los efectos principales. Una interacción significativa frecuentemente oscurece la significación de los efectos principales. Cuando hay interacción significativa, se deberán examinar los niveles de un factor, digamos A, con los niveles del o de los otros factores fijos, para tener conclusiones sobre el efecto principal A.

3.7 Introducción a los métodos multivariantes

3.7.1 Análisis de Componentes Principales

En muchas ocasiones los datos son de “*alta dimensión*” es decir la información que se dispone corresponde a la medida de un cierto número de variables (que pueden ser desde una decena a varios miles) sobre los sujetos del estudio. Dado que es imposible discernir tendencias mediante una inspección visual de dicha matriz, es necesario reducir la dimensión para permitir el análisis visual. Como el análisis visual se realiza tradicionalmente en dos dimensiones, en un sistema de coordenadas de x e y , muchos métodos permiten la reducción de una matriz de cualquier dimensión a sólo dos dimensiones.

Si queremos mostrar los datos en sólo dos dimensiones, deberemos ser capaces de capturar la mayor cantidad de la variabilidad de los datos que sea posible en tan sólo estas nuevas dos dimensiones. El *Análisis de Componentes Principales (PCA)* ha sido desarrollado con este propósito.

La información que se puede obtener de un análisis de componentes principales depende de si existe una tendencia discernible en los datos que se pueda ver reflejada en dos dimensiones. Como usualmente las matrices de datos tendrán más de 3 variables (único caso en el podríamos representar las observaciones para ver si es factible encontrar las nuevas componentes), será imposible saber “*a priori*” si existe esta tendencia. Para conocer la ”calidad” de la reducción se calcula el porcentaje de variabilidad retenido por cada componente, en principio para que la reducción de la dimensión sea efectiva las dos primeras componentes tendrían que retener un porcentaje superior al 50%. Podemos aplicar diferentes criterios a la hora de decidir con cuántas componentes nos quedamos.

- 1) Uno puede ser a partir del valor de la proporción de variabilidad total de cada componente. Se puede fijar un nivel mínimo y quedarnos con el número de componentes necesario para superar este valor mínimo.
- 2) El segundo puede ser que una componente no puede tener una desviación estándar menor que una de las variables originales. Si hemos escalado cada variable original dividiendo por su desviación estándar entonces la desviación estándar de cada componente ha de ser mayor que uno.
- 3) Otro criterio puede ser ver en qué momento se produce un descenso de la desviación estándar muy notable. Quedarnos con las componentes previas.

Las nuevas componentes se obtienen como una combinación lineal de las variables originales. En muchas ocasiones es difícil encontrar el significado de estas componentes, como variables compuestas, por lo que el uso principal de la técnica es la reducción de la dimensión como paso previo a la aplicación de otros análisis posteriores, por ejemplo, un diagrama de dispersión de las primeras componentes

con el objeto de encontrar agupaciones en los datos o con el objeto de contrastar similitudes o diferencias entre los individuos.

```
> golub.pca = prcomp(golub, scale = TRUE, center = TRUE)
> summary(golub.pca)
> plot(golub.pca)
```

El argumento center=TRUE centra los datos restando la media de la columna de modo que las variables tengan medias nulas. El argumento scale=TRUE hace que las variables originales sean divididas por su desviación estándar de modo que la varianza (y la desviación estándar) de las nuevas variables sea la unidad.

3.7.2 Análisis de conglomerados (Cluster Analysis)

El objetivo del *Análisis Cluster* es obtener grupos de objetos de forma que, por un lado, los objetos pertenecientes a un mismo grupo sean muy semejantes entre sí, es decir, que el grupo esté cohesionado internamente y, por el otro, los objetos pertenecientes a grupos diferentes tengan un comportamiento distinto con respecto a las variables analizadas, es decir, que cada grupo esté aislado externamente de los demás grupos. Una vez establecidas las variables y los objetos a clasificar el siguiente paso consiste en establecer una medida de proximidad o de distancia entre ellos que cuantifique el grado de similaridad entre cada par de objetos.

Las *medidas de proximidad*, similitud o semejanza miden el grado de semejanza entre dos objetos de forma que, cuanto mayor es su valor, mayor es el grado de similaridad existente entre ellos y con más probabilidad los métodos de clasificación tenderán a ponerlos en el mismo grupo. Cuando las variables son cuantitativas una de las medidas que se puede utilizar es el coeficiente de correlación de Pearson.

Las *medidas de disimilaridad*, desemejanza o distancia miden la distancia entre dos objetos de forma que, cuanto mayor sea su valor, más diferentes son los objetos y menor la probabilidad de que los métodos de clasificación los pongan en el mismo grupo. Como las medidas de distancia son sensibles a la diferencia de escalas o de magnitudes hechas entre variables es necesaria la estandarización de datos, cuando éstos corresponden a variables cuantitativas, para evitar que las variables con una gran dispersión tengan un mayor efecto en la similaridad. Cuando las variables que se miden son cuantitativas una de las medidas de distancia más populares es la distancia euclídea. Si se han medido p variables a cada objeto la distancia euclídea se calcula como:

$$d_{ij} = \sum_{k=1}^p (X_{ik} - X_{jk})^2$$

Métodos jerárquicos

En los métodos jerárquicos, inicialmente cada objeto a agrupar es un grupo en sí mismo y sucesivamente se van fusionando grupos cercanos hasta que todos los individuos confluyen en un solo grupo. Los métodos jerárquicos pueden dividirse en aglomerativos y divisivos. En los primeros se parte de tantas clases como objetos tengamos que clasificar y en pasos sucesivos vamos obteniendo clases de objetos similares, mientras que en los segundos se parte de una única clase formada por todos los objetos que se va dividiendo en grupos más pequeños sucesivamente. La principal ventaja de los métodos jerárquicos aglomerativos es que se puede representar el problema en forma de árbol o dendograma donde se observa muy bien la solución final.

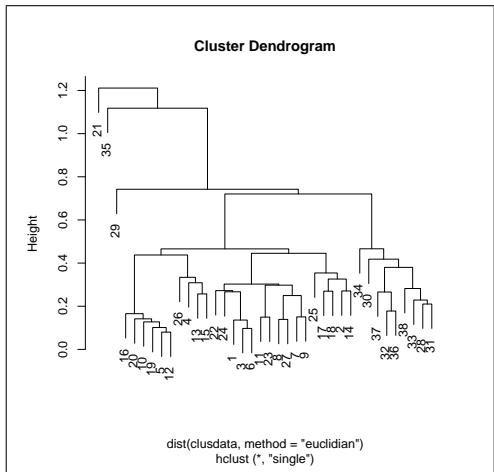
Un algoritmo básico aglomerativo es:

- Calcular la matriz de distancias entre los objetos.
- Identificar cada objeto con un clúster.
- Combinar los dos grupos más cercanos y actualizar la matriz de distancias.
- Repetir el paso anterior hasta que sólo quede un grupo.

Como ejemplo usaremos los datos de Golub et al. (1999). Se sospecha que la expresión de los genes "CCND3 CyclinD3" y "Zyxin" difieren entre los grupos **ALL** y **AML**. Vamos a estudiar si, en nuestros datos, las expresiones de estos dos genes permiten confirmar la afirmación anterior.

```
> clusdata<-data.frame(golub[1042,],golub[2124,])
> colnames(clusdata)<-c("CCND3 Cyclin
+ D3", "Zyxin")
> plot(hclust(dist(clusdata,method="euclidean"), method="single"))
```

Figura 25. Dendrograma



En la figura puede observarse la formación de dos clusters.

Método K-means

En el *Método K-means*, no es necesario realizar el cálculo inicial de las distancias entre todos los objetos. Existen varias formas de implementarlo pero todas ellas siguen, básicamente, los siguientes pasos:

- Se seleccionan k centroides o semillas donde k es el número de grupos deseado.
- Se asigna cada observación al grupo cuya semilla es la más cercana.
- Se calculan los puntos semillas o centroides de cada grupo.
- Se iteran los 2 pasos anteriores hasta que se satisfaga un criterio de parada como, por ejemplo, los puntos semillas apenas cambian o los grupos obtenidos en dos iteraciones consecutivas son los mismos.

El método suele ser muy sensible a la solución inicial dada por lo que es conveniente utilizar una que sea buena. Una forma de construirla es a partir de una clasificación obtenida por un algoritmo jerárquico.

Para exemplificar un método de análisis *k-means* simularemos dos cadenas de expresión génica para dos poblaciones normales. Es decir, tomaremos al azar 50 expresiones génicas de 2 poblaciones normales $N(0; 0.5)$ y 50 expresiones génicas de dos poblaciones normales $N(2; 0.5)$. Los puntos de datos se recogen en dos matrices de (50x2) que se unen colocando una encima de la otra. Sobre el total de cien puntos de datos se lleva a cabo un análisis de conglomerados con $k = 2$.

```
> data <-rbind(matrix(rnorm(100,0,0.5), ncol = 2), matrix(rnorm(100,2,0.5), ncol = 2))
> cl<-kmeans(data, 2) #K-means clustering with 2 clusters of sizes 50, 50
```

Part II

Análisis de datos de microarrays

4. El proceso de análisis de datos de microarray (MDA)

4.1 Introducción

Este capítulo es una corta transición entre la primera parte del curso, en la que se han presentado los conceptos y herramientas básicos y la segunda en donde se presentan por separado y con mayor detalle los métodos de análisis de datos de microarrays.

Su objetivo por tanto es ofrecer una visión *de conjunto* que sirva de guía (“roadmap”) para los capítulos siguientes de forma que sin perder el detalle de cada uno de ellos tengamos conciencia de en qué punto del proceso general nos encontramos.

El capítulo se estructura en dos partes. En la primera se presentan brevemente algunos de los problemas que típicamente se puede querer estudiar con microarrays u otras técnicas similares de análisis de datos de alto rendimiento. A continuación se presenta lo que se ha llamado aquí el *proceso de análisis de microarrays*. Finalmente se introducen algunos casos reales que, a modo de ejemplo se utilizarán en los capítulos siguientes.

4.2 Tipos de estudios

Los microarrays y otras tecnologías de alto rendimiento se han aplicado a multitud de investigaciones, de tipos muy diversos que van desde estudio del cáncer ([1, 24, 50]) al de la germinación y la maduración del tomate ([39]). A pesar de ello no resulta complicado clasificar los estudios realizados en algunos de los grandes bloques que se describen a continuación. La clasificación está basada en el excelente texto de Simon y colegas ([46]) y aunque se origina en problemas de microarrays se puede aplicar fácilmente a estudios de genómica o ultrasecuenciación.

4.2.1 Comparación de grupos o *Class comparison*

El objetivo de los estudios comparativos es determinar si los perfiles de expresión génica difieren entre grupos previamente identificados. También se conoce estos estudios como de *selección de genes diferencialmente expresados* y son, sin duda los más habituales. Los grupos pueden representar una gran variedad de condiciones, desde distintos tejidos a distintos tratamientos o múltiples combinaciones de factores experimentales.

El análisis de este tipo de experimentos, que se describe en el capítulo sobre selección de genes diferencialmente expresados utiliza herramientas estadísticas como las pruebas de comparación de grupos paramétricas (t de Student) o no (test de Mann-Whitney) o diversos métodos de análisis de la varianza.

Entre los ejemplos de la sección 4.3 los casos 4.3.1, 4.3.3 o 4.3.4 hacen referencia a estudios comparativos.

4.2.2 Predicción de clase o *Class prediction*

La predicción de clase puede confundirse con la selección de genes en tanto que disponemos de clases predefinidas pero su objetivo es distinto, ya que no pretende simplemente buscar genes cuya expresión sea distinta entre dichos grupos sino genes que puedan ser utilizados para identificar a qué clase pertenece un “nuevo” individuo dado cuya clase es “a priori” desconocida. El proceso de predicción suele empezar con una selección de genes informativos, que pueden ser, o no, los mismos que se obtendrían si aplicáramos los métodos del apartado anterior, seguida de la construcción de un modelo de predicción y, lo que es más importante, de la verificación o validación de dicho modelo con unos datos nuevos independientes de los utilizados para el desarrollo del modelo.

Aunque el interés de la predicción de clase es muy alto se trata de un procedimiento mucho más complejo y con más posibilidades de error que la simple selección de genes diferencialmente expresados.

Entre los ejemplos de la sección 4.3 el caso 4.3.2 trata de un problema de predicción, a la vez que uno de descubrimiento de clases.

4.2.3 Descubrimiento de clases o *Class discovery*

Un problema distinto a los descritos se presenta cuando no se conoce las clases en que se agrupan los individuos. En este caso de lo que se trata es de encontrar grupos entre los datos que permitan reunir a los individuos más parecidos entre sí y distintos de los de los demás grupos. Los métodos estadísticos que se emplearan en estos casos se conocen como *análisis de clusters* y no son tan complejos como los de predicción de clase aunque algunos aspectos como por ejemplo la definición del número de grupos no resulta tampoco sencillo.

Entre los ejemplos de la sección 4.3 tanto el caso `golub`, en parte, como el `??` tratan problemas de descubrimiento de clases.

Una curiosidad del campo de la estadística es que el término clasificación aparece usado de forma indistinta para referirse a problemas de predicción de clase o de descubrimiento de clase.

4.2.4 Otros tipos de estudios

Una vez identificados los principales tipos de estudios quedan muchos que no coinciden plenamente con ninguno de ellos. Sin entrar en detalles podemos señalar los estudios de evolución a lo largo del tiempo (“time course”), los de significación biológica (“Gene Enrichment Analysis”, “Gene Set Enrichment Analysis”, …), los que buscan relaciones entre los genes (“network analysis” o “pathway analysis”). De momento con conocer e identificar los tres grandes bloques mencionados resultará más que suficiente.

4.3 Algunos ejemplos concretos

Una de las dificultades con que se encuentra la persona que comienza en el análisis de datos de microarrays es de donde obtener ejemplos concretos con los que practicar las técnicas que está aprendiendo.

No es difícil encontrar datos de microarrays en internet por lo que se han seleccionado algunos conjuntos de datos interesantes para utilizarlos de ejemplo a lo largo del curso. Algunos de éstos son “populares” en el sentido de que han sido utilizados en diversas ocasiones y por lo tanto se encuentran bien documentados. Otros se han escogido simplemente porque ilustran bien algunas de las ideas que se desea exponer o por su accesibilidad.

Todos los datos corresponden a investigaciones publicadas por lo que no se describen exhaustivamente sinó que se expone brevemente el origen y objetivos del trabajo –incluyendo su clasificación según los grupos definidos en la sección anterior– y las características perinentes para el análisis como el tipo de microarrays, los grupos –si los hay– o como acceder a los datos.

4.3.1 Estudio de procesos regulados por citoquinas

Efecto de la estimulación con LPS sobre los procesos regulados por citoquinas

Este conjunto de datos, que se denominará “celltypes”, corresponde a un estudio realizado por Chelvarajan y sus colegas ([8]) que analizaron el efecto de la estimulación con lipopolisacáridos en la regulación por parte de citoquinas de ciertos procesos biológicos relacionados con la inflamación.

Este estudio es del tipo “class comparison” es decir su principal objetivo es la obtención de genes diferencialmente expresados entre dos o más condiciones.

Los datos se encuentran disponibles en la base de datos pública **caarray** mantenida por el *National Institute of Health (NIH)*, pero pueden descargarse de la página de materiales del curso para garantizar su disponibilidad.

4.3.2 Clasificación molecular de la leucemia

Clasificación molecular para distinguir variantes de leucemia mieloblástica aguda

A finales de los años 90, Todd Golub y sus colaboradores ([24]) realizaron uno de los estudios más populares hasta el momento con datos de microarrays. En él utilizaron microarrays de oligonucleótidos para 6817 genes humanos para mirar de encontrar una forma de distinguir (clasificar) tumores de pacientes con leucemia linfoblástica aguda (ALL) de aquellos que sufrían de leucemia mieloide aguda (AML). Además se interesaba por la posibilidad de descubrir subgrupos de forma que pudieran definirse variantes de cada una de estas patologías a nivel molecular.

La diversidad de objetivos del estudio lleva a clasificarlo tanto entre los del tipo de predicción de clase como entre los que buscan descubrir nuevas clases o grupos en los datos.

Los datos de este estudio se encuentran disponibles en la web del instituto Broad, en donde se llevó a cabo (<http://www.broadinstitute.org>). También se encuentra disponible un paquete de R denominado **ALL** que permite utilizarlos directamente usando R y Bioconductor.

4.3.3 Efecto del estrógeno y el tiempo de administración

Efecto del tratamiento con estrógenos en la expresión de genes relacionados con cáncer de mama

Scholtens y colegas ([45]) describen un estudio sobre el efecto de un tratamiento con estrógenos y del tiempo transcurrido desde el tratamiento en la expresión génica en pacientes de cáncer de mama. Los investigadores supusieron que los genes asociados con una respuesta temprana podrían considerarse dianas directas del tratamiento, mientras que los que tardaron más en hacerlo podrían considerarse objetivos secundarios correspondientes a dianas más alejadas en las vías metabólicas.

Estos datos han sido utilizados multitud de veces en los cursos de análisis de microarrays realizados por el proyecto Bioconductor y se encuentran disponibles en forma de paquete de R, el paquete **estrogen**. Una característica importante de este paquetes es el hecho de que en vez de los datos procesados proporciona los datos “crudos” en forma de archivos .CEL de Affymetrix. Esto permite una mayor flexibilidad a la hora de reutilizarlos lo que explica su popularidad.

4.3.4 Efecto del CCL4 en la expresión génica

Efecto del tratamiento con dimetilsulfóxido (DMSO) en la expresión génica

Holger y colegas de la empresa LGC Ltd. en Teddington, Inglaterra realizaron unos experimentos con microarrays de dos colores en los que trataron hepatocitos de ratón con tetraclorido de carbono (CCL4) o con dimetilsulfóxido (DMSO). El tetraclorido de carbono fue ampliamente utilizado en productos de limpieza o refrigeración para el hogar hasta que se detectó que podía tener efectos tóxicos e incluso cancerígenos. El DMSO es un solvente similar, sin efectos tóxicos conocidos, que se utilizó como control negativo.

Los datos de este estudio no han sido publicados pero se encuentran disponibles en el paquete CCL4 de bioconductor. Su interés reside por un lado en que se trata de datos de microarrays de dos colores de la marca Agilent –en un momento en que la mayoría de estudios se realizan con datos de un color. Aparte de esto cabe resaltar el hecho de que el paquete incluye, de forma similar al anterior, los datos “crudos” en forma de archivos de tipo “Genepix” uno de los programas populares para escanear imágenes generadas con microarrays de dos colores.

Este estudio es también un estudio de comparación de clases cuyo objetivo principal es la selección de genes cuya expresión se asocia al tratamiento con CCL4 o DMSO.

4.3.5 Análisis de patrones en el ciclo celular

Busqueda de patrones de coexpresión en datos de ciclo celular de levadura Los datos de este ejemplo denominado **kidney** son datos ya normalizados referidos a la expresión de los genes en distintos momentos del ciclo celular de la levadura e decir desde que concluye la división celular hasta que se inicia la siguiente.

Los datos pueden descargarse desde la página del proyecto “Yeast Cell Cycle Project” (Proyecto de estudio del ciclo celular de la levadura) en la dirección:
<http://genome-www.stanford.edu/cellcycle/data/rawdata/>.

4.3.6 Recapitulación

La tabla 6 resume la lista de ejemplos que se utilizan en este manual indicando el nombre con que nos referiremos de aquí en adelante a cada conjunto de datos así como algunas de sus características.

4.4 El proceso de análisis de microarrays

Una vez descritos los tipos de análisis y algunos ejemplos podemos pasar a describir el proceso de análisis de microarrays que se resume brevemente en la figura 26.

Tabla 6. Conjuntos de datos utilizados en este manual. Aparte del nombre (arbitrario y “mnemotécnico”) se indica el tipo de microarrays, el número de muestras, y el tipo de problema para el que se utilizaron originalmente.

Nombre	Tipo	N. Muestras	Tipo de estudio
<code>celltypes</code>	Un color (Affy, Mouse 4302)	12	Comparativo
<code>golub</code>	Un color (Affy, HGU95A)	38	Clasificación
<code>estrogen</code>	Un color (Affy, HGU95A)	8	Comparativo
<code>CCL4</code>	Dos colores (Agilent, WG Rat Microarray)	16	Comparativo
<code>breastTum</code>	Un color (Affy, HGU95A)	49	Clasificación

El análisis de microarrays, como la mayoría de análisis debe proceder de forma ordenada y siguiendo el método científico:

- La pregunta y su contexto nos servirán de guía para definir el *Diseño experimental* adecuado.
- El experimento se deberá realizar siguiendo las pautas decididas en el *Diseño experimental* y los datos obtenidos –que solemos denominar datos “crudos” o “raw data”– deberán someterse a los *Controles de calidad adecuados* antes de continuar con su análisis.
- Una vez decidido si la calidad de los datos es aceptable pasaremos a prepararlos para el análisis lo que puede incluir diversas formas de *preprocesado*, o *transformaciones* que a menudo se incluyen de forma general bajo el paraguas del término *normalización*, aunque, como veremos se trata de conceptos distintos.
- Los datos normalizados se utilizarán para los *análisis estadísticos* que hayamos decidido realizar durante el diseño experimental.
- Finalmente los resultados de los análisis serán la base para una *interpretación biológica* de los resultados del experimento.

Figura 26. Proceso de análisis de microarrays

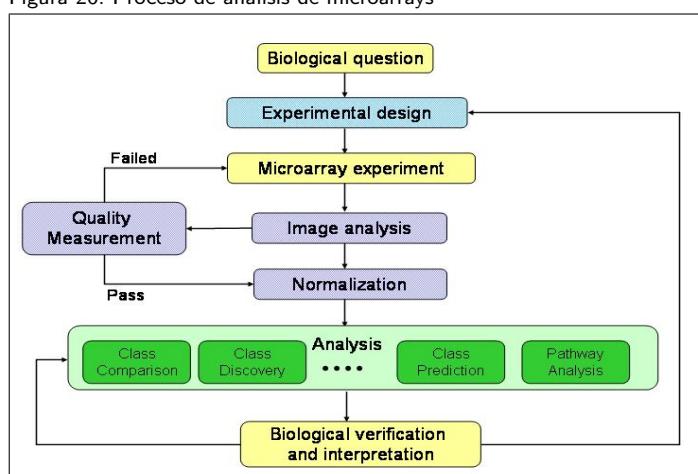


Figura 26

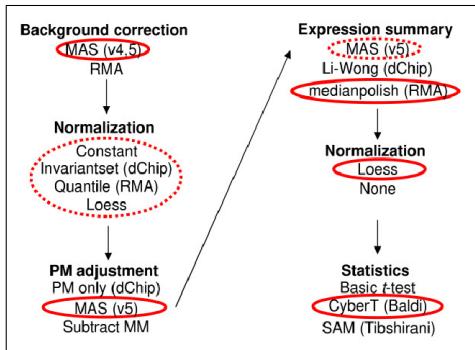
El análisis de microarrays puede ser fácilmente visualizado como un proceso que empieza por una pregunta biológica y concluye con una interpretación de los resultados de los análisis que, de alguna forma, confiamos nos acerque un poco a la respuesta de la pregunta inicial.

El proceso descrito es básicamente una forma razonable de proceder en general. Los microarrays y otros datos genómicos son diferentes en su naturaleza de los datos clásicos alrededor de los que se han desarrollado la mayor parte de técnicas estadísticas. En consecuencia, en muchos casos ha sido necesario adaptar las técnicas existentes o desarrollar otras nuevas para adecuarse a las nuevas situaciones encontradas. Esto ha determinado que existan muchos métodos para cada una de los pasos descritos anteriormente lo que da lugar a una grandísima cantidad de posibilidades.

En la práctica lo que suele hacerse es optar por utilizar algunos de los métodos en los que hay un cierto consenso acerca de su calidad y utilidad para cada problema. Allison ([2]) repasa los puntos principales de este consenso dando una lista de puntos a tener en cuenta en cualquier estudio que utilice microarrays. Imbeaud y Auffray ([27]) citan una lista de hasta 39 puntos que uno debe seguir en un experimento con microarrays para usar “buenas prácticas”.

Finalmente Zhu y otros ([55]) utilizan un conjunto de arrays con valores de expresión conocidos para proponer los que, a su parecer, resultan los métodos más apropiados para cada etapa desde la corrección del background hasta la selección de genes diferencialmente expresados. La figura 27 ilustra algunas de las opciones sugeridas por dichos autores.

Figura 27. Diseño de arrays.



Los capítulos que siguen al presente proceden aproximadamente en el orden del proceso descrito en 27. Se empieza por tratar los principios del diseño de experimentos. A continuación se describen algunos métodos para el control de calidad, el preprocessado y la normalización de los datos. Se sigue con los métodos de selección de genes –adaptados de los métodos descritos en el capítulo ??, y los métodos de clasificación, para concluir con una introducción a los métodos de análisis de la significación biológica de las listas de genes obtenidas de los procesos anteriores.

5. Diseño de experimentos de microarrays

En este capítulo se examinan un componentes clave del análisis de microarrays el diseño de experimentos que, no tan sólo es crucial para una buena recogida de información sino que marca todo el proceso desde el preprocesado al análisis final.

5.1 Fuentes de variabilidad

Los datos genómicos son muy variables.

Figura 28. Proceso de análisis de microarrays

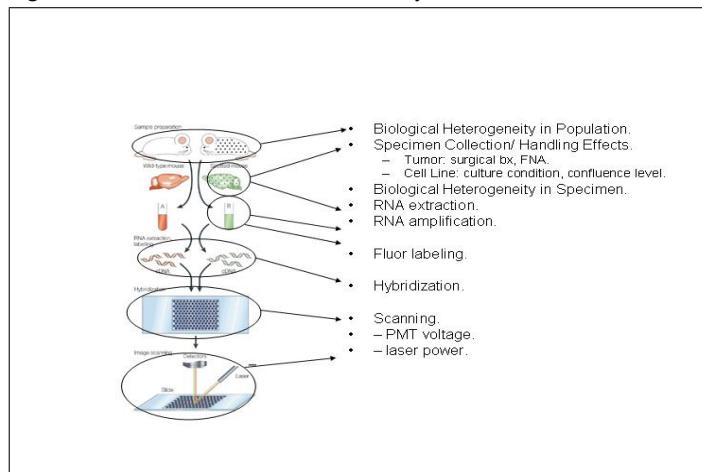


Figura 28

Geschwind ([23]) ilustra algunas posibles fuentes de variabilidad, desde que se inicia el experimento hasta que se lee la información con el scanner.

Habitualmente, en la mayor parte de situaciones experimentales, podemos distinguir entre variabilidad sistemática y aleatoria.

La variabilidad sistemática es principalmente debida a procesos técnicos mientras que la variabilidad aleatoria es atribuible tanto a razones técnicas como biológicas. Podemos encontrar ejemplos de variabilidad sistemática en la extracción del ARN, marcaje o fotodetección. La variabilidad aleatoria puede estar relacionada con muchos factores tales como la calidad del ADN o las características biológicas de las muestras.

La manera natural de manejar la variabilidad aleatoria es, por supuesto, la utilización de un diseño experimental apropiado y el apoyo para obtener conclusiones de unas herramientas estadísticas adecuadas. Los problemas relacionados con el diseño de los experimentos los discutiremos en esta sección y los relacionados con la aplicación de métodos estadísticos serán tratados en la sección ??.

Tradicionalmente, se estiman las correcciones de la variabilidad sistemática a partir de los datos, en lo que se llama genéricamente "calibrado". En este contexto, hablaremos de "normalización", que se tratará en la sección ??.

5.2 Principales conceptos en Diseño de Experimentos

Vamos a plantear alguna definiciones de conceptos que aparecen de forma usual cuando se plantea un diseño:

- **Unidad experimental:** Entidad física a la que se aplica un tratamiento, de forma independiente al resto de unidades. En cada unidad experimental se pueden realizar una medida o varias medidas, en este caso distinguiremos entre unidades experimentales y unidades observacionales. .
- **Factor:** son las variables independientes que pueden influir en la variabilidad de la variable de interés.
- **Factor tratamiento:** es un factor del que interesa conocer su influencia en la respuesta.
- **Factor bloque:** es un factor en el que no se está interesado en conocer su influencia en la respuesta pero se supone que ésta existe y se quiere controlar para disminuir la variabilidad residual.
- **Niveles:** cada uno de los resultados de un factor. Según sean elegidos por el experimentador o elegidos al azar de una amplia población se denominan factores de efectos fijos o factores de efectos aleatorios.
- **Tratamiento:** es una combinación específica de los niveles de los factores en estudio. Son, por tanto, las condiciones experimentales que se desean comparar en el experimento. En un diseño con un único factor son los distintos niveles del factor y en un diseño con varios factores son las distintas combinaciones de niveles de los factores.
- **Tamaño del Experimento:** es el número total de observaciones recogidas en el diseño.

5.3 Principios básicos en el diseño del experimento

Al planificar un experimento, aparte de la replicación, hay tres principios básicos que se deben tener siempre en cuenta: La replicación, el control local o "bloqueo" y la *aleatorización*. Aplicados correctamente dichos principios garantizan diseños experimentales eficientes.

5.4 Replicación

La replicación o repetición de un experimento de forma idéntica en un número determinado de unidades, es la que permite la realización de un posterior análisis estadístico. La utilización de réplicas es importante para incrementar la precisión, obtener suficiente potencia en los tests y como base para los procedimientos de inferencia. Normalmente, distinguimos dos tipos de réplicas en el análisis de microarrays:

- *La replicación técnica* se utiliza cuando estamos tratando réplicas del mismo material biológico. Puede ser tanto la replicación de spots en el mismo chip como la de diferentes alicuotas de la misma muestra hibridadas en diferentes microarrays.
- *Replicación biológica* se da cuando se toman medidas de múltiples muestras.

La replicación técnica permite la estimación del error a nivel de medida, mientras que la replicación biológica permite estimar la variabilidad a nivel de población.

Figura 29.

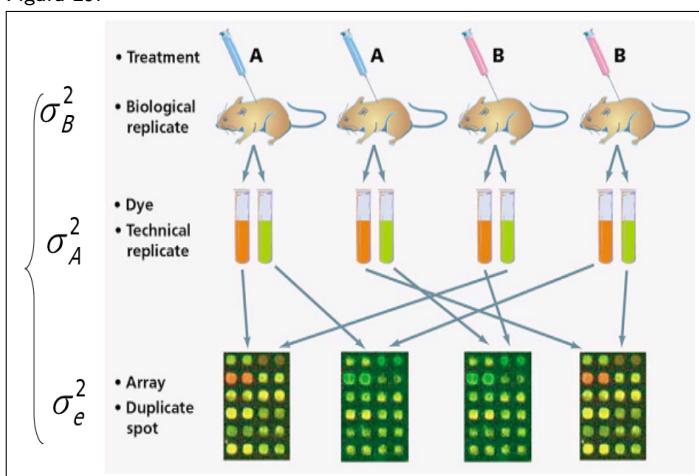


Figura 29

Réplicas biológicas vs técnicas

5.4.1 Potencia y tamaño de la muestra

Sorprendentemente, los primeros experimentos de microarrays utilizaban pocas o ninguna réplica biológica. La principal explicación para este hecho - además de la falta de conocimiento estadístico - fué el alto coste de cada microarray. En pocos años, la necesidad de las réplicas ha llegado a ser indiscutible, y al mismo tiempo, el coste de los chips ha disminuido considerablemente. Actualmente, es normal la utilización de, al menos, tres a cinco réplicas por condición experimental, aunque a este consenso se ha llegado más por razones empíricas que por la disponibilidad de modelos apropiados para el análisis de la potencia y tamaño de la muestra.

Recientemente, se ha producido una importante afluencia de artículos describiendo métodos para el análisis de la potencia y tamaño de la muestra. A pesar de

su variedad, ningún método aparece como candidato claro para ser utilizado en situaciones prácticas. Esto se debe, probablemente, a la complejidad de los datos de microarrays, básicamente por que los genes no son independientes. Por tanto, estas estructuras de correlación existen en los datos, pero la mayor parte de las dependencias son desconocidas por lo que la estimación de estas estructuras es muy complicada.

Como indicaba Allison ([2]) aunque no hay consenso sobre que procedimiento es mejor para determinar el tamaño de las muestras, sí que lo hay sobre la conveniencia de realizar el análisis de la potencia, y, por supuesto, sobre el hecho de que un mayor número de replicas generalmente proporcionan una mayor potencia.

5.4.2 Pooling

En el contexto de los microarrays, llamamos "pooling" a la combinación de mARN de diferentes casos en una única muestra. Hay dos razones a favor de ello, una, es que, a veces, no hay suficiente ARN disponible y esta es la única forma de conseguir suficiente material para construir los arrays, otra, más controvertida, es la creencia que la variabilidad entre arrays puede reducirse por "pooling". La justificación es que combinar muestras equivale a promediar expresiones, y como ya se conoce, el promedio es menos variable que los valores individuales. A pesar de la debilidad de este argumento, es verdad que en ciertas situaciones el pooling puede ser apropiado y, recientemente, muchos estadísticos han dedicado sus esfuerzos a tratar de responder la pregunta "to pool or not to pool" ([30]). Por ejemplo, si la variabilidad biológica está altamente relacionada con el error en las medidas, y las muestras biológicas tienen un coste mínimo en comparación con el de los arrays, una apropiada estrategia de "pooling" puede ser claramente eficiente en costes.

De todos modos, el "pooling" no se debería usar en cualquier tipo de estudios. Si el objetivo es comparar expresiones medias (ver más adelante "comparación de clases"), puede funcionar adecuadamente, pero se debería claramente evitar cuando el objetivo del experimento es construir predictores que se basen en características individuales.

Aleatorización

Se entiende por aleatorizar la asignación de todos los factores al azar a las unidades experimentales. Co ello se consigue disminuir el efecto de los factores no controlados por el experimentador en el diseño experimental y que podrían influir en los resultados.

Las ventajas de aleatorizar los factores no controlados son:

- Transforma la variabilidad sistemática no planificada en variabilidad no planificada o ruido aleatorio. Dicho de otra forma, aleatorizar previene contra la introducción de sesgos en el experimento.
- Evita la dependencia entre observaciones al aleatorizar los instantes de recogida muestral.
- Valida muchos de los procedimientos estadísticos más comunes.

Bloquear

Hace referencia a dividir o particionar las unidades experimentales en grupos llamados bloques de modo que las observaciones realizadas en cada bloque se realicen bajo condiciones experimentales lo más parecidas posibles. A diferencia de lo que ocurre con los factores tratamiento, el experimentador no está interesado en investigar las posibles diferencias de la respuesta entre los niveles de los factores bloque.

Bloquear es una buena estrategia siempre y cuando sea posible dividir las unidades experimentales en grupos de unidades similares. La ventaja de bloquear un factor que se supone que tiene una clara influencia en la respuesta pero en el que no se está interesado, es que convierte la variabilidad sistemática no planificada en variabilidad sistemática planificada.

5.5 Diseños experimentales para microarrays de dos colores

En arrays de dos colores, se aplican dos condiciones experimentales a cada array. Esto permite la estimación del efecto del array, como el efecto de bloqueo. En Affymetrix u otro array de un canal, cada condición se aplica a un chip separado, imposibilitando la estimación del efecto de los microarrays, lo cual, por otra parte, se considera que tiene una relación muy pequeña en el tratamiento de los efectos debido al proceso industrial utilizado para fabricar estos chips.

Como consecuencia de lo anterior, los experimentos que usan arrays de un canal se consideran “estandar”, por lo que se les pueden aplicar los conceptos y técnicas tradicionales de diseño de experimentos .

Los arrays de dos canales presentan una situación más complicada. Por una parte los “dos colores” no son simétricos, es decir, con la misma cantidad de material, un array hibrido con uno u otro color, sea Cy5 o Cy3, emitirá señales con diferente intensidad. La forma normal de manejar este problema es *dye-swapping* que consiste en utilizar para una misma comparación dos arrays con dyes cambiados, es decir, si en el primer array la muestra 1 se marca con Cy3 y la muestra 2 con Cy5, con las muestras del segundo array se hace al revés (ver

figura 30).

Figura 30.

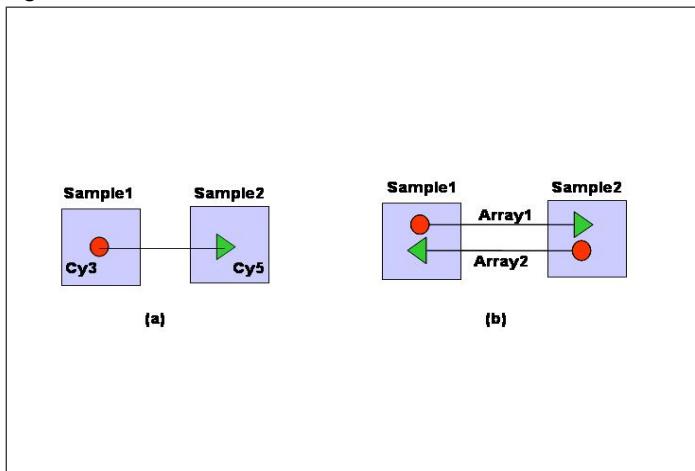


Figura 30

(a) Representación simplificada de un diseño. Cada flecha representa un array de dos canales en el que el origen indica el marcaje con Cy3. (b) Dye swapping.

Por otra parte, el hecho de que solo se puedan aplicar dos condiciones a cada array complica el diseño, ya sea porque normalmente hay más de dos condiciones, o porque no es recomendable hibridar directamente dos muestras en un array, creando pares artificiales.

El problema de la asignación eficiente de muestras a microarrays, dado un número de condiciones a comparar y un número fijo de arrays disponibles, ha sido estudiado de forma exhaustiva.

El diseño utilizado más comúnmente en la comunidad biológica es el *diseño de referencia* en el que cada condición de interés se compara con muestras tomadas de alguna referencia estandar común a todos los arrays. (ver figura 31 (a)).

Figura 31. Diseño de arrays.

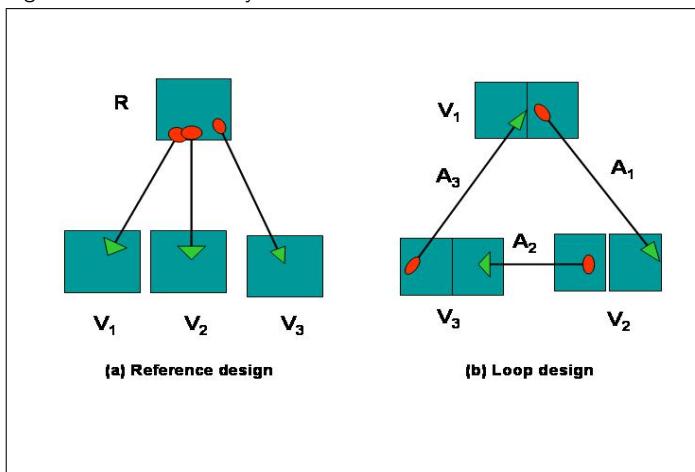


Figura 31

(a) Diseño de referencia. (b) Diseño en loop.

Los diseños de referencia permiten hacer comparaciones indirectas entre condiciones de interés. La critica más importante a esta aproximación es que el 50% de las fuentes de hibridación se utilizan para producir la señal del grupo control, de un interés no intrínseco para los biólogos. En contraste, un diseño en loop compara dos condiciones a través de otra cadena de condiciones, por lo que elimina

la necesidad de una muestra de referencia (ver figura 31(b)).

6. Exploración de los datos, control de calidad y preprocesado

6.1 Introducción

Los estudios realizados con microarrays, sea cual sea la tecnología en que se basan, tienen una característica común: generan grandes cantidades de datos a través de una serie de procesos, que hacen que su significado no siempre sea completamente intuitivo.

Como en todo tipo de análisis, antes de empezar a trabajar con los datos, debemos de asegurarnos de que éstos son fiables y completos y de que se encuentran en la escala apropiada para proporcionar la información que pretendemos obtener de ellos.

En el caso de los microarrays solemos distinguir dos fases previas al análisis de los datos:

1) **Exploración y control de calidad** Mediante gráficos y/o resúmenes numéricos estudiamos la estructura de los datos con el fin de decidir si parecen correctos o presentan anomalías que deben ser corregidas.

2) **Preprocesado** Incluso si son correctos, los datos “crudos” no sirven para el análisis sino que necesitan ser...

- “Limpiados”, para eliminar la parte de la señal no atribuible a la expresión, llamada “background” o ruído.
- “Normalizados” para hacerlos comparables entre muestras y eliminar posibles sesgos técnicos.
- “Resumidos” o “sumarizados” de forma que se tenga un sólo valor por gen.
- “Transformados” de forma que la escala sea razonable y facilite el análisis.

A menudo –por un cierto abuso de lenguaje– se denomina “normalización” al preprocesado descrito en las etapas anteriores.

6.1.1 Nivel de análisis y tipo de microarray

Desde la generalización del uso de los microarrays se han desarrollado muchas formas para visualizar los datos y decidir acerca de su calidad. Algunas trabajan

sobre los datos obtenidos del escáner, otras lo hacen con los datos normalizados. Algunas sirven tanto para arrays de dos colores como arrays de un color. Otras son específicas de la tecnología. Con el fin de organizar esta multitud de opciones podemos diferenciar:

- **Datos de bajo nivel** Son los datos proporcionados por el escáner contenidos por ejemplo en archivos .gpr (dos colores) o .cel (un color). Estos últimos son binarios por lo que no es posible ni tan sólo visualizarlos sin programas específicos.
- **Datos de alto nivel** Son los datos resultantes del (pre)procesado de los datos de bajo nivel. Básicamente se corresponden con los datos de expresión ya resumidos (“sumarizados”), normalizados o no.

La exploración y el control de calidad pueden basarse en datos de bajo o de alto nivel. La tabla 7 muestra una clasificación de los distintos procedimientos según el tipo de datos y el tipo de microarray. Dichos procedimientos se explican en las secciones siguientes.

Tabla 7. Procedimientos de visualización y control de calidad según el tipo de datos (de bajo o alto nivel) y el tipo de microarrays (de dos colores o un color)

Tipo de microarrays Nivel de análisis	General	Dos colores	Un color
Bajo Nivel	PCA	R-G Scatterplots	Imagen (Sondas)
1 color: (sondas)	Histograma	MA-plot	Gr. de degradación
2 colores: (1 canal R o G)	Boxplot	Signal2Noise plot	Gr. de densidad
		Imagen (R o G)	Gr. de “probesets”
Alto nivel	PCA	Imagen (R)	MA-Plots
1 color: Expresión absoluta	Histogramas	Imagen (G)	PLM-plots
2 colores: Expresión relativa	Boxplots	Imagen (R/G)	(NUSE, RLE, Residuos)

6.1.2 Datos de partida

La estructura de los datos de microarrays de un color o de dos colores difiere considerablemente, tanto a nivel físico (los chips y las imágenes que de ellos se obtiene) como informático, es decir en la forma en que se representan.

Arrays de dos colores (o de “cDNA”)

Tradicionalmente los arrays de dos colores o de cDNA se realizaban de forma menos automatizada que los de un color o de Affymetrix. Esto implica que, tras obtener la imagen, el escaneado del archivo “.TIFF”¹ resultante pudiera ser ll-

¹ .TIFF es un formato para archivos de imagen de alta calidad

evado a cabo mediante un *software* independiente como **Genepix**. Este programa convierte las imágenes en números y genera un archivo de información (con extensión “.gpr”) a partir del cual pueden calcularse las expresiones relativas, así como valores de calidad para cada *spot* o punto escaneado en la imagen.

Para cada imagen (o sea para cada microarray) hay un archivo .gpr que contiene una fila por gen y varias columnas con distintos valores, por ejemplo la intensidad para cada canal, valores resumen de las intensidades y controles de calidad (“FLAGS”).

La tabla 8 muestra lo que serían las primeras filas y columnas de un archivo “.gpr” obtenido mediante el programa “genepix”.

Tabla 8. Ejemplo simplificado de las primeras filas y columnas de un archivo “.gpr”

Gen	Señal	Fondo	Señal	Fondo	“Flag”	Otros
	Cy5 (R)	Cy5 (Rb)	Cy3 (G)	Cy3 (Gb)		
gen-1	3.7547	1.8128	5.0672	1.8496	1	...
gen-2	0.8331	0.9175	1.1536	0.9995	0	...
gen-3	9.8254	0.2781	0.6921	0.5430	1	...
gen-4	9.1539	0.1918	3.8290	0.0014	0	...
gen-5	4.8603	0.2377	0.5338	0.3335	0	...
gen-6	7.8567	1.3941	1.3050	1.4876	1	...
gen-7	2.7619	0.8108	8.4916	1.6518	0	...
gen-8	3.6618	0.1918	0.3770	1.1842	0	...
gen-9	4.4300	0.5888	2.8161	0.8707	1	...
...

Los valores de intensidad se convierten en una única *matriz de expresión* que contiene una columna por chip con los valores de intensidad relativa obtenidas por ejemplo con una “sumarización” del tipo:

$$\log \frac{R - Rb}{G - Gb}$$

y una fila por gen (mismas filas que archivos .gpr).

La tabla 9 muestra lo que sería una matriz de expresión derivada de cuatro archivos .gpr como los de la tabla 8.

Arrays de un color (Affymetrix)

El resultado de escanear la imagen de un array de affymetrix es un archivo de extensión “.CEL” que, a diferencia de los arrays de dos colores, está en formato binario es decir que solo puede ser leído con programas específicos para ello.

Tabla 9. Ejemplo simplificado de las primeras filas y columnas de una matriz de expresión relativa obtenida de cuatro archivos “.gpr”

	Array-1	Array-2	Array-3	Array-4
gen-1	1.65695	-0.10820	1.69515	8.25137
gen-2	-1.82305	0.11350	1.58807	0.95676
gen-3	0.01561	0.47682	-27.88036	-1.39078
gen-4	0.42709	4.29319	-4.31366	30.96866
gen-5	0.04332	0.24126	-10.27675	0.26680
gen-6	-0.02825	0.68408	2.04163	0.99554
gen-7	3.50549	-8.04635	0.18286	0.22647
gen-8	-0.23261	-1.08477	0.19582	1.11561
gen-9	0.50643	1.52147	0.99092	0.23441
...

De forma similar a los arrays de dos colores, existe un archivo .CEL por cada microarray chip, que contiene los valores PM (*perfect match*) y MM (*mismatch*) para cada sonda.

A partir de las intensidades de los archivos .CEL se genera la matriz de expresión que contiene una columna por chip con los valores de intensidad absoluta y una fila por grupo de sondas. En el caso de arrays de affymetrix existe una gran variedad de algoritmos de “sumarización” y según cual se utilice se obtendrá unos u otros valores de expresión pero estos serán siempre medidas absolutas, es decir independientes del resto de muestras.

La tabla 10 muestra las primeras filas de una matriz de expresión sumarizada correspondiente a los primeros genes del caso resuelto 1.

Tabla 10. Ejemplo simplificado de las primeras filas y columnas de una matriz de expresión absoluta obtenida de cuatro archivos “.cel”

ID_REF	GSM188013	GSM188014	GSM188016	GSM188018
1007_s_at	15630.2	17048.8	13667.5	15138.8
1053_at	3614.4	3563.22	2604.65	1945.71
117_at	1032.67	1164.15	510.692	5061.2
121_at	5917.8	6826.67	4562.44	5870.13
1255_g_at	224.525	395.025	207.087	164.835
...

Datos de ejemplo

Los ejemplos de este capítulo se basarán en dos de los conjuntos de datos descritos en el capítulo 4.

- Por un lado los datos del conjunto **estrogen** referidos al efecto del tratamiento con estrógenos en cancer de mama descrito en 4.3.3. La exploración se basará en los datos normalizados pero también en los datos “crudos”, de bajo nivel, contenidos en los archivos “.cel”. Estos datos pueden obtenerse directamente del paquete de Bioconductor **estrogen**. Un aspecto interesante de este conjunto de datos es que, junto con 8 muestras “buenas” se proporciona un array defectuoso, denominado “**bad.cel**” que facilita el ver como aparecen ciertos gráficos cuando hay problemas.
- Los datos del conjunto **CC14** relativos al efecto del tratamiento con CCL4 en la expresión de los hepatocitos. Los datos resultan accesibles al instalar el paquete **CC14**.

Los ejemplos de este capítulo, como los del resto del manual son el resultado de ejecutar las instrucciones adecuadas de R. Con el fin de no romper la continuidad del texto en general no se mostrarán dichas instrucciones –o al menos no de forma que se puedan reproducir. En los apéndices se proporcionará el código que permite reproducir cualquier ejemplo o cálculo llevado a cabo.

6.2 Gráficos para la exploración y control de calidad

Los gráficos son útiles para comprobar la calidad de los datos de microarrays, obtener información sobre cómo se deben preprocesar los datos y comprobar, finalmente, que el preprocesado se haya realizado correctamente.

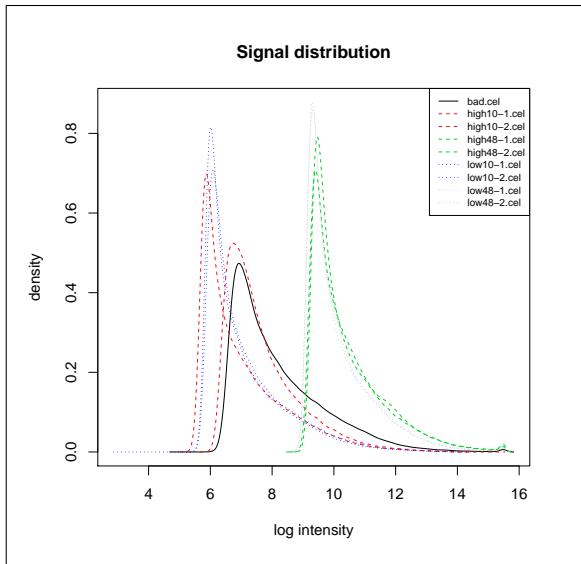
Siguiendo el esquema presentado en la tabla ?? se presentan a continuación los distintos gráficos utilizados con una breve descripción de lo que representa cada uno y como interpretarlos adecuadamente. El código para generarlos se presenta al final del capítulo como un apéndice.

6.2.1 Control de calidad con gráficos estadísticos generales

Histogramas y gráficos de densidad

Estos gráficos permite hacerse una idea de si las distribuciones de los distintos arrays son similares en forma y posición. La figura 32 muestra los histogramas correspondientes a los 9 arrays del conjunto de datos **estrogen**.

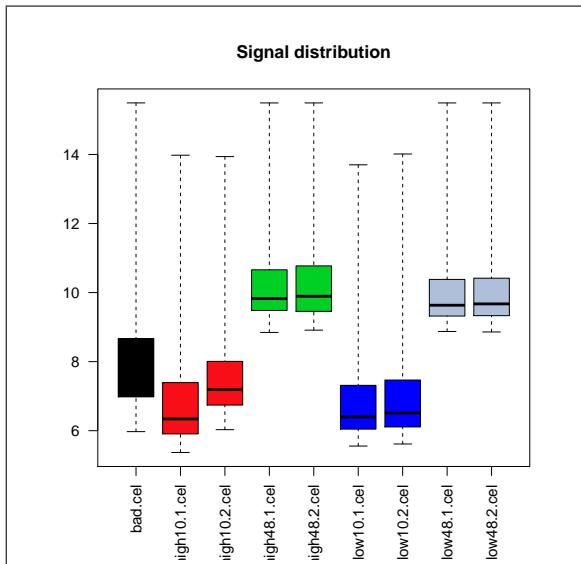
Figura 32. Histograma de los arrays correspondientes a los datos de estrogen.



Diagramas de caja o “boxplots”

Como los histogramas los diagramas de caja –basados en los distintos cuantiles de las valores– dan una idea de la distribución de las intensidades. La figura 33 muestra los diagramas de caja correspondientes a los 9 arrays del conjunto de datos **estrogen**.

Figura 33. Diagramas de cajas de los arrays correspondientes a los datos de estrogen.



Gráficos de componentes principales

El análisis de componentes principales puede servir para detectar si las muestras se agrupan de forma “natural” es decir con otras muestras provenientes del mismo grupo o si no hay correspondencia clara entre grupos experimentales y proximidad en este gráfico. Cuando esto sucede no significa necesariamente que haya un problema pero puede ser indicativo de efectos técnicos –como el conocido efecto

“batch”– que podría ser necesario corregir.

```
> plotPCA <- function ( X, labels=NULL, colors=NULL, dataDesc="", scale=FALSE)
+ {
+   pcX<-prcomp(t(X), scale=scale) # o prcomp(t(X))
+   loads<- round(pcX$sdev^2/sum(pcX$sdev^2)*100,1)
+   xlab<-c(paste("PC1",loads[1],"%"))
+   ylab<-c(paste("PC2",loads[2],"%"))
+   if (is.null(colors)) colors=1
+   plot(pcX$x[,1:2],xlab=xlab,ylab=ylab, col=colors,
+         xlim=c(min(pcX$x[,1])-10, max(pcX$x[,1])+10),
+         ylim=c(min(pcX$x[,2])-10, max(pcX$x[,2])+10),
+         )
+   text(pcX$x[,1],pcX$x[,2], labels, pos=3, cex=0.8)
+   title(paste("Plot of first 2 PCs for expressions in", dataDesc, sep=" "), cex=0.8)
+ }
```

La figura 34 muestra dos diagramas de componentes principales realizados a partir de los datos normalizados del conjunto de datos **estrogen**. El gráfico de la parte superior que incluye el array defectuoso ilustra que la principal fuente de variabilidad es la diferencia de este array con el resto. Cuando se repite el análisis omitiendo esta muestra puede verse como la principal fuente de variación (eje X) se asocia con el tiempo de exposición (alto a la derecha, bajo (10h) a la izquierda, mientras que la segunda fuente de variación se asocia con la exposición a los estrógenos (alto arriba, bajo abajo).

Figura 34. Gráfico de las dos primeras componentes principales para los datos de **estrogen**.

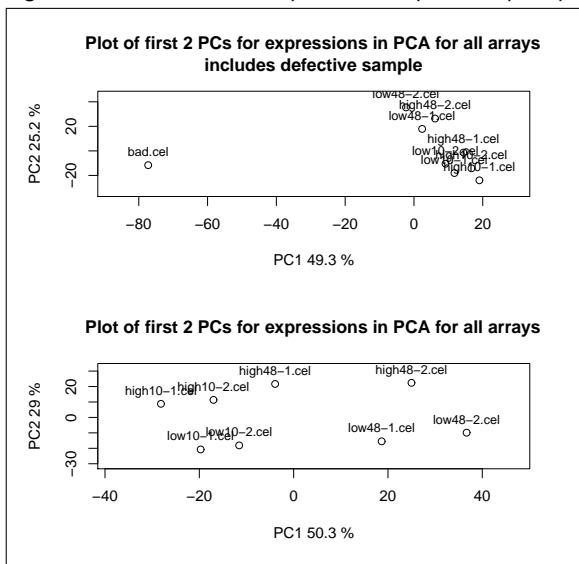


Imagen del chip

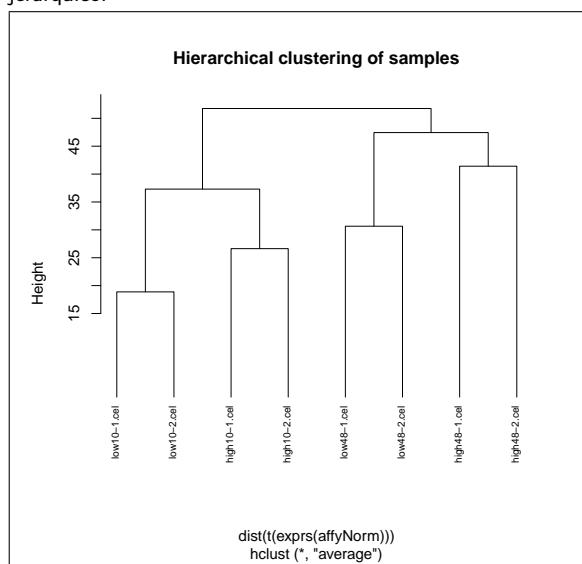
Otra forma de ver si las muestras se agrupan según los grupos experimentales, o mediante otros criterios es usando un cluster jerárquico que realiza una agru-

pación básica de las muestras por grado de similaridad según la distancia que se utilice.

Como en el caso de las componentes principales si las muestras se agrupan según las condiciones experimentales es una buena señal pero si no es así puede deberse a la presencia de otra fuente de variación o bien al hecho de que se trata de un gráfico basado en todo los datos y las condiciones experimentales pueden haber afectado un pequeño número de genes.

La figura 35 muestra como se agrupan los datos del conjunto **estrogen** en base a un cluster jerárquico. Como en el caso de las componentes principales tras eliminar el array defectuoso las muestras se separan, primero por el tiempo de exposición y luego por niveles de estrógeno suministrado.

Figura 35. Agrupación de las muestras para los datos de **estrogen** en base a un cluster jerárquico.



6.2.2 Gráficos de diagnóstico para microarrays de dos colores

El diagnóstico de arrays de dos canales se basa principalmente en la imagen y en diferentes tipos de gráficos.

Diagramas de dispersión y “MA-plots”

La normalización discutida en este mismo capítulo es un punto clave en el proceso de análisis de microarrays y se ha dedicado un gran esfuerzo a desarrollar y probar diferentes métodos ([41, 53]). Una razón para ello es que hay diferentes artefactos técnicos que deben ser corregidos para poder ser utilizados, y no cualquier método puede funcionar con todos ellos.

En general, los métodos de normalización se basan en el siguiente principio:

La mayor parte de los genes en un array se pueden expresar o no expresar ante cualquier condición, pero se espera que sólo una pequeña cantidad de genes muestre cambios de expresión entre condiciones.

Esto da una idea de como debería ser un gráfico de intensidades. Por ejemplo, si no hubiese artefactos técnicos, en arrays de dos canales, una gráfica de dispersión de intensidad del rojo frente al verde debería dejar la mayor parte de los puntos alrededor de una diagonal. Cualquier desviación de esta situación debería ser atribuible a razones técnicas, no biológicas, y por tanto, debería ser eliminada. Esto ha conducido a un método de normalización muy popular consistente en estimar la transformación a aplicar, como una función de las intensidades utilizando el método *lowess* en la representación transformada de la gráfica de dispersión conocida como el *gráfico MA-plot*.

La figura 36 (a) muestra un gráfico de dispersión del canal rojo frente al verde en un array de dos colores. El hecho de que los datos no estén centrados alrededor de la diagonal sugiere la necesidad de normalización.

Una representación muy popular que ayuda a visualizar mejor esta asimetría es lo que se conoce como “*MA-plot*” (36(b)). Geométricamente representa una rotación del gráfico de dispersión, en la que el significado de los nuevos ejes es:

- $A = \frac{1}{2}(\log_2(R * G))$: El logaritmo de la intensidad media de los dos canales,
- $M = \log_2 \frac{R}{G}$: El logaritmo de la expresión relativa entre ambos canales (normalmente conocido como “log-ratio”).

Figura 36. (a) Gráfico de un canal frente al otro (b) MA-plot (intensidad frente “log-ratio”)

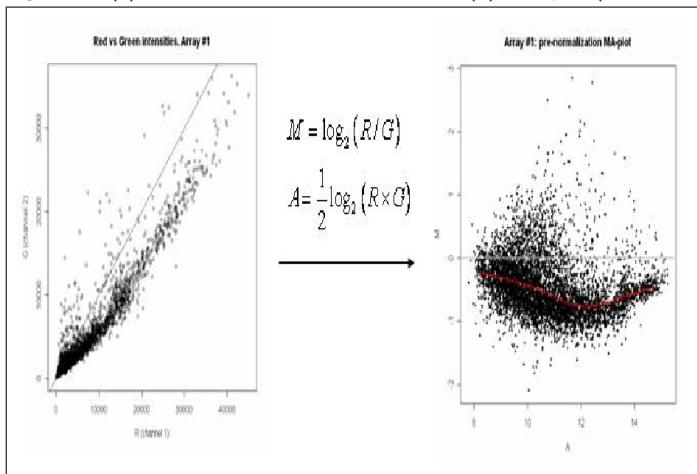


Imagen del array

La imagen del chip (véase la figura 37, izquierda) ofrece una visión rápida de la calidad del array, proporcionando información acerca del balance del color, la uniformidad en la hibridación y en los *spots*, de si el background es mayor

del normal y de la existencia de artefactos como el polvo o pequeñas marcas (rasguños).

Histogramas de señales y de la relación señal–ruído

Estos gráficos (véase la figura 37, derecha) son útiles para detectar posibles anomalías o un background excesivamente alto.

Figura 37. Imágenes de buena calidad, como se puede ver en el gráfico superior, deberían tener un background bajo y una alta relación señal/ruido. Imágenes de mala calidad, gráfico inferior, muestran un background alto y baja relación señal/ruido.

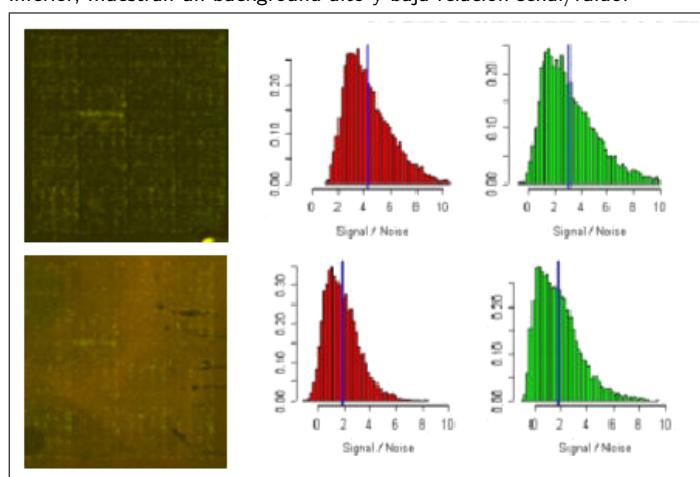


Figura 37

Útil para detectar posibles anomalías o un background excesivamente alto

Boxplots

Un gráfico muy utilizado es el diagrama de cajas o “boxplot” múltiple con una caja por cada chip. Del alineamiento (o falta de él) y la semejanza (o disparidad) entre las cajas, se deduce si hace falta, o no, normalizar entre arrays.

En el caso de arrays de dos colores pueden utilizarse diagramas de cajas “dentro de arrays” (entre distintos sectores del mismo chip) y “entre arrays”.

6.2.3 Gráficos de diagnóstico para microarrays de un color

Imagen del chip

Los arrays de affymetrix contienen millones de sondas por lo que no pueden examinarse a simple vista. A pesar de ello hay diversas formas de obtener una imagen que, en caso de presentar irregularidades pueden indicar algún tipo de problemas como burbujas, arañazos, etc. La figura 38 muestra algunas de las imágenes

Figura 38. Imágenes de cuatro microarrays de Affymetrix.

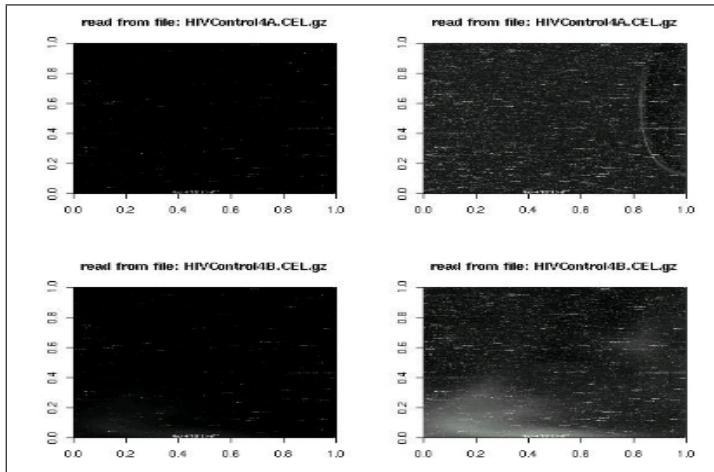
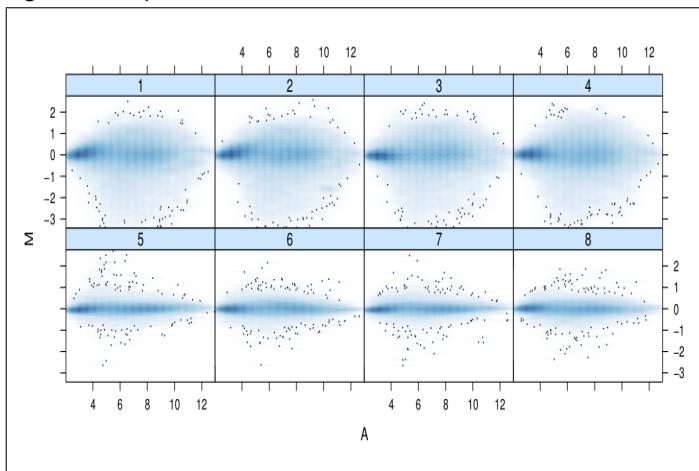


Gráfico "M-A"

En los chips de dos colores el MA-plot se utiliza para comparar los dos canales en cada array (rojo y verde). En cambio, en los chips de Affymetrics, en que sólo hay un canal en cada array, la única forma de definir M (el log ratio) es a partir de la comparación entre pares de de valores, ya sea los arrays dos a dos o bien cada array respecto un valor de referencia que puede ser la mediana, punto a punto, de todos los arrays (véase por ejemplo la figura 39).

Figura 39. MAplot de un canal



$M = \log_2(I_1) - \log_2(I_2)$: log ratio $A = \frac{1}{2}(\log_2(I_1) + \log_2(I_2))$: log de intensidades
Donde I_1 es la intensidad del array de estudio, e I_2 es la intensidad media de arrays. Por lo general, se espera que la distribución en el gráfico se concentre a lo largo del eje $M = 0$.

Figura 38

La imagen del array de Affymetrix sólo es útil para evidenciar grandes problemas como burbujas, arañazos, etc. En este ejemplo los dos arrays de la izquierda se considerarían aceptables y los de la derecha defectuosos.

Figura 39

En los chips de Affymetrix la única forma de definir M (el log ratio) es comparar entre diferentes arrays

Gráficos de degradación

La calidad de la hibridación del ARN a lo largo de los conjuntos de sondas puede variar. Si un grupo de arrays el nivel de degradación es desigual suele considerarse que su calidad es baja.

La figura 40 muestra los gráficos de degradación para el conjunto de datos **estrogen** pudiendo observarse como, en este caso el array **bad.cel** no presenta un patrón de degradación distinto.

Figura 40.

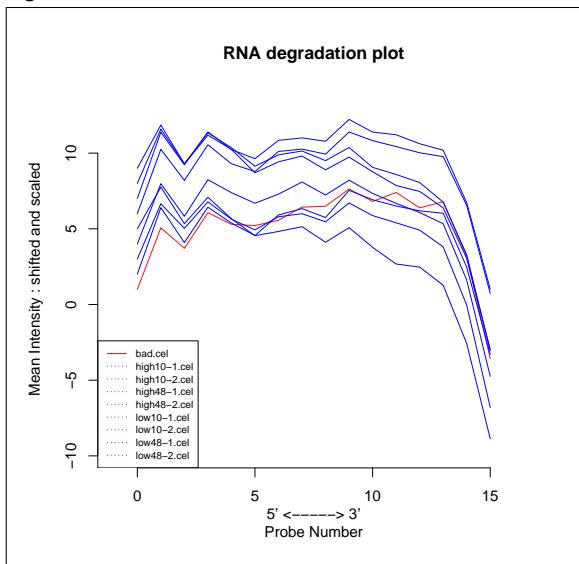


Figura 40

Gráficos de diagnóstico de densidad y de degradación

Modelos de bajo nivel (“Probe-Level-Models” o PLM)

Los modelos de bajo nivel (“Probe-Level-Models” o PLM) ajustan a los valores de intensidad –a nivel de sondas, no de valores totalizados de gen– un modelo explicativo. Los valores estimados por este modelo se comparan con los valores reales y se obtienen los errores o “residuos” del ajuste. El análisis de dichos residuos procede de forma similar a lo que se realiza al analizar un modelo de regresión: Si los errores no presentan ningún patrón especial supondremos que el modelo se ajusta relativamente bien. Si, en cambio, observamos desviaciones de esta presunta aleatoriedad querrá decir que el modelo no explica bien las observaciones, lo cual se atribuirá a la existencia de algún problema con los datos.

Con los valores ajustados del modelo se calculan dos medidas:

- La expresión relativa en escala logarítmica “ Relative Log Expression” (RLE) es una medida estandarizada de la expresión. No es de gran utilidad pero debería presentar una distribución similar en todos los arrays.
- El error no estandarizado y normalizado o “NUSE” es el más informativo ya que representa la distribución de los residuos a la que hacíamos referencia

más arriba. Si un array es problemático la caja correspondiente en el boxplot aparece desplazada hacia arriba o abajo de las demás.

La figura 41 muestra los gráficos RLE y NUSE para el conjunto de datos estrogen. En ambos gráficos puede verse como el array defectuoso “bad.cel” queda claramente diferenciado del resto.

Figura 41.

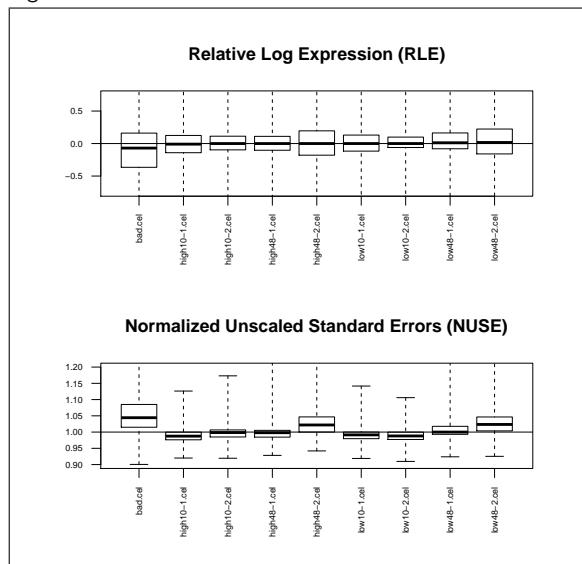


Figura 41

Gráficos de diagnóstico calculados a nivel de sondas (“PLM”)

6.3 Normalización de arrays de dos colores

La palabra *normalización* describe las técnicas utilizadas para transformar adecuadamente los datos antes de que sean analizados. El objetivo es corregir diferencias sistemáticas entre muestras, en la misma o entre imágenes, lo que no representa una verdadera variación entre las muestras biológicas.

Estas diferencias sistemáticas pueden deberse, entre otras, a:

- Cambios en la tinción que producen sesgos la intensidad del *spot*.
- La ubicación en el array que puede afectar el proceso de lectura.
- Un problema en la placa de origen.
- La existencia de diferencias en la calidad de la impresión: pueden presentarse variaciones en los pins y el tiempo de impresión
- Cambio en los parámetros de la digitalización (escaneo).

A veces puede ser difícil detectar estos problemas, aunque existen algunas formas de saber si es necesaria realizar una normalización. Aquí destacamos dos posibilidades:

- 1) Realizar una auto-hibridación. Si hibridamos una muestra con ella misma, las intensidades deberían ser las mismas en ambos canales. Cualquier desviación de esta igualdad, significa que hay un sesgo sistemático.
- 2) Detectar artefactos espaciales en la imagen o en la tinción de los gráficos de diagnóstico

6.3.1 Normalización global

Este método está basado en un ajuste global, es decir en modificar todos los valores una cantidad c , estimada de acuerdo a algún criterio.

$$\log_2 R/G \rightarrow \log_2 R/G - c = \log_2 R/(Gk) \quad (4)$$

Opciones para k o $c = \log_2 k$ son

c = mediana o media de log ratio para un conjunto concreto de genes o genes control o genes housekeeping.

La intensidad total de la normalización

$$k = \sum R_i / \sum G_i$$

6.3.2 Normalización dependiente de la intensidad

En este caso se realiza una modificación específica para cada valor. Esta modificación se obtiene como una función de la intensidad total del gen ($c = c(A)$).

$$\log_2 R/G \rightarrow \log_2 R/G - c(A) = \log_2 R/(Gk(A)) \quad (5)$$

Una posible estimación de esta función puede hacerse utilizando la función *lowess* (LOcally WEighted Scatterplot Smoothing).

6.4 Resumen y normalización de microarrays de Affymetrix

En los arrays de Affymetrix, como en todos los tipos de microarrays, tras escanear la imagen se obtiene una serie de valores de intensidad de cada elemento del chip. En el caso de estos arrays sabemos que cada valor no corresponde a la expresión de un gen:

- Hay múltiples valores (sondas o "probes") por cada gen, que originan un *probe-set*.
- Cada grupo de sondas consiste en múltiples pares de sondas, donde cada puede tener dos elementos:

Un "perfect match" que coincide exactamente con el fragmento del gen al que corresponde la sonda

Un "mismatch" que coincide con el anterior salvo por el valor central que se ha sustituido por el nucleótido complementario. Estos "mismatches" se introdujeron en los primeros arrays de affymetrix para tener una medida de hibridación no específica pero en las versiones más recientes se han abandonado.

El proceso que convierte las señales individuales en valores de expresión normalizados para cada gen consta de tres etapas:

- 1) Corrección del ruido de fondo o "background"
- 2) Normalización para hacer los valores comparables
- 3) "Sumarización"(Resumen) o concentración de los valores de cada grupo de sondas en un único valor de expresión absoluto normalizado para cada gen.

A menudo los tres pasos se denominan genérica -y erróneamente- "normalización".

A diferencia de los chips de ADNc, aquí las medidas de expresión son absolutas (no se compara una condición contra otra) dado que cada chip se hibrida con una muestra.

Hay muchos métodos para estimar la expresión (más de 30 publicados). Cada método contempla de forma explícita o implícita las tres formas de preprocesado: corrección del fondo, normalización y resumen.

Los principales métodos que consideraremos son:

- Microarray Suite (MAS). Método oficial de Affymetrix. Versiones 4.0 y 5.0
- dChip: Li and Wong. Método basado en modelos multichip.
- RMA (Bioconductor). Actualmente es el método "estándar".

6.4.1 Métodos originales de Affymetrix

M.A.S. 4.0

Es el primer método introducida por Affymetrix. La corrección del fondo se realiza restando el "perfect match" del "mismatch"

$$E_j = PM_j - MM_j \quad (6)$$

La normalización se realiza de forma global haciendo transformaciones de forma que la media de todo el chip sea la misma y la summarización se basa en calcular el promedio de las diferencias absolutas ignorando los pares que se desvían más de 3σ de μ .

$$Dif.Media = \frac{1}{|A|} \sum_{j \in A} (PM_j - MM_j) \quad (7)$$

Los problemas que presenta este método son:

- 1/3 de los MM son mayores que los PM
- Pueden aparecer valores MM negativos
- El uso de los MM añade ruido

M.A.S. 5.0

Algunos problemas que presentaba el método M.A.S. 4.0 llevaron a sustituirlo por otra variante, el M.A.S 5.0, llamado así por venir implementado en el software de affymetrix llamado "MicroArray Suite 5.0".

Este método utiliza un estadístico robusto, *el biweight de Tukey*, para corregir y ponderar el fondo y calcular (estimar) la señal. El biweight de Tukey T_{bi} pondera los valores por su distancia a la mediana, es decir, mide la tendencia central pero realiza un ajuste de outliers.

La lógica de este método reside en pensar que el valor de MM no siempre tiene sentido, (p.ej si MM > PM). Dado que esto sucede en ocasiones se realiza el cambio siguiente:

- 1) Se introduce el background específico de un conjunto de pruebas i de tamaño

n basado en los pares de pruebas j :

$$SB_i = T_{bi}(\log(PM_{i,j}) - \log(MM_{i,j})) : j = 1, \dots, n \quad (8)$$

2) SB se utiliza para decidir como se ajusta el background

- Si es grande los datos suelen ser fiables
- Si es pequeño mejor basarse tan solo en PM

Este método no tan solo corrige el background sino que también permite normalizar y sumarizar. Para ello se introduce el "Mismatch Idealizado" (IM) que permite corregir la intensidad de las pruebas individuales. Este método también ha sido muy criticado:

- Se considera que no tiene mucho sentido promediar las pruebas entre arrays, pues éstos pueden tener características de hibridación intrínsecamente distintas.
- El método no mejora "aprendiendo" del funcionamiento entre arrays de las pruebas individuales.

6.4.2 El método RMA (Robust Multi-Array Average)

Para compensar algunas deficiencias de los primeros métodos de resumen y normalización de arrays de Affymetrix, Irizarry y sus colegas introdujeron en 2003 ([28]) un método basado en la modelización de las intensidades de las sondas que, en vez de basarse en las distintas sondas de un gen dentro de un mismo array se basa en los distintos valores de la misma sonda entre todos los arrays disponibles,

Esquemáticamente los pasos que realiza este método son:

- 1) Ajusta el ruido de fondo (background) basándose solo en los valores PM y utilizando un modelo estadístico complejo en el que combina la modelización de la señal mediante una distribución exponencial con la del ruido mediante una distribución normal.
- 2) Toma logaritmos base 2 de cada intensidad ajustada por el background.
- 3) Realiza una *normalización por cuantiles* de los valores del paso 2 consistente en substituir cada valor individual por el que tendría la misma posición en la distribución empírica estimada sobre todas las muestras, es decir los promedios de las distribuciones de los valores ordenados de cada array (véase figura 42)

4) Estima las intensidades de cada gen separadamente para cada conjunto de sondas. Para ello realiza una técnica similar a una regresión robusta denominada "median polish" sobre una matriz de datos que tiene los arrays en filas y los grupos de sondas en columnas.

Como resultado final de todos los pasos anteriores se obtiene la matriz con los datos sumarizados y normalizados. A pesar de no estar exento de críticas como la que afirma que este procedimiento "compacta" los valores reduciendo su variabilidad natural, este método se ha convertido en el estándar "de facto" actualmente por muchos usuarios de Bioconductor.

Figura 42. Esquema de la normalización por cuantiles.

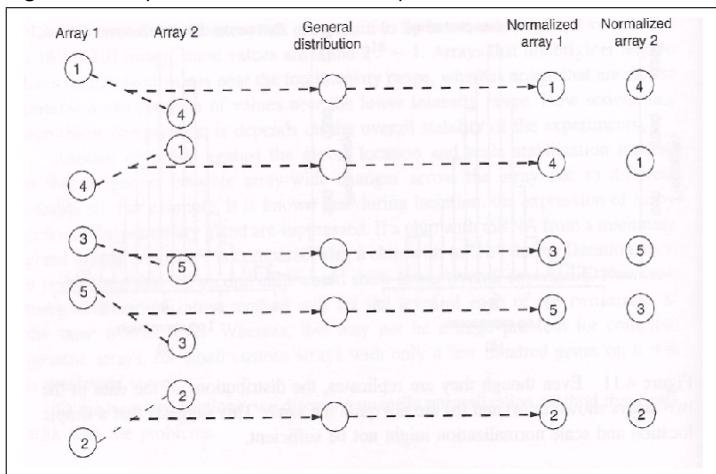


Figura 42

El método RMA incluye una normalización por cuantiles como la representada esquemáticamente en esta figura

6.5 Filtraje no específico

El filtraje no específico es recomendable para eliminar el ruido de fondo y limitar los ajustes posteriores a los necesarios. Los principales procesos de filtrado son:

- Eliminación de los spots marcados como erróneos mediante flags y que son debidos a problemas en la hibridación o en el escaneo.
- Eliminación de spots con señales muy bajas debido a problemas en el *spotting* o a que no ha habido hibridación en ese spot (Filtraje pr.).
- Eliminación de genes que no presenten una variación significativa en su señal, entre distintas condiciones experimentales (Filtraje por variabilidad). Ante la duda, *se debe ser conservador y reducir la operación de filtraje al mínimo*

El objetivo del filtraje es eliminar aquellos spots cuyas imágenes o señales sean erróneas por diferentes motivos, disminuyendo el ruido de fondo. Aunque existe controversia a su uso, prefiriendo el no filtrado a eliminar de forma no intencionada spots informativos.

7. Selección de genes diferencialmente expresados

7.1 Introducción

El motivo más habitual para el que se suelen utilizar microarrays es la búsqueda de genes cuya expresión cambia entre dos o más condiciones experimentales, por ejemplo a consecuencia de un tratamiento, una enfermedad u otras causas (distintos tiempos, distintas líneas celulares, ...).

El problema consiste en identificar estos genes y suele denominarse *selección de genes diferencialmente expresados* (“DEG”) o bien comparación de clases.

El problema de seleccionar genes diferencialmente expresados se traduce de manera casi inmediata al problema estadístico de comparar variables y, en años recientes, se han desarrollado un gran número de métodos estadísticos para resolverlo. La mayoría son extensiones de los métodos estadísticos clásicos –pruebas t o análisis de la varianza– adaptados en uno u otro sentido para tener en cuenta las peculiaridades de los microarrays.

Aunque el problema de la selección de genes diferencialmente expresados puede relacionarse directamente con la realización de pruebas estadísticas, en el caso de los microarrays, el hecho de que haya dos tecnologías que miden la expresión de dos formas distintas hace que se deba diferenciar la metodología a emplear en cada caso. Los arrays de dos colores combinan dos muestras en un chip y generan una medida de expresión relativa. Esto hace que para comparar dos muestras de un mismo individuo sean la opción naturalmente más apropiada.

En el caso de querer comparar muestras independientes de diferentes individuos los arrays de un color son la mejor opción. Evidentemente, lo más común será disponer de una sola técnica y tener que adaptar los análisis estadísticos a la misma.

Vamos a plantear un posible esquema de trabajo, para situar la mejor opción en cada caso:

- Situación 1: experimento con 5 individuos diabéticos y 5 no diabéticos, independientes entre si (muestras independientes)
 - Caso 1: Arrays de cDNA (2 colores): Utilizaríamos 5 arrays Diabético/Referencia y 5 arrays No diabético/Referencia

- Caso 2: Arays de Affymetrix (1 color): Utilizaríamos 5 arrays de Diabético y 5 arrays de No diabético
- Situación 2: experimento con 6 individuos de los que se ha tomado una muestra de tejido sano y otra de tejido tumoral (muestras apareadas o dependientes)
- Caso 3: Arrays de cDNA (2 colores): Utilizaríamos 6 arrays, uno por individuo, y en cada uno se realizaría la comparaci'on Tejido Tumoral/Tejido sano.
- Caso 4: Arays de Affymetrix (1 color): Utilizaríamos 12 arrays, 2 por individuo, 6 con muestras de Tejido Tumoral y 6 con muestras de Tejido sano.

7.1.1 Medidas *naturales* para comparar dos muestras

Recordemos que una vez se han hecho los experimentos con microarrays y NA valor detectado por el escaner. Esto hace que algunas operaciones que se realicen tengan en cuenta esta característica. Segun si la comparación a realizar se llevará a cabo con datos *independientes* (2 muestras, casos 1 y 2) o con datos *dependientes* (muestras apareadas, casos 3, 4) algunas medidas *naturales* o razonables para la comparación de expresiones son las siguientes:

- Para comparaciones directas, con expresiones relativas entre muestras apareadas o bien diferencias apareadas de expresiones absolutas:
 - log ratio promedio : $\bar{R} = \frac{1}{n} \sum_{i=1}^n R_i$
 - t-test de una muestra $\frac{\bar{R}}{SE}$, donde SE estima el error estándar del *log ratio* promedio
 - t-test robusto: Substituir en el anterior medidas robustas del error estándar
- Para comparaciones indirectas entre muestras independientes de expresiones relativas o absolutas:
 - Diferencia media $\bar{R}_1 - \bar{R}_2 = \frac{1}{n_1} \sum_{i=1}^{n_1} R_i - \frac{1}{n_2} \sum_{j=1}^{n_2} R_j$
 - t-test (clásico) de dos muestras $\frac{\bar{R}_1 - \bar{R}_2}{SE_{12} \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$
 - t-test robusto de dos muestras: Substituir en el anterior medidas robustas del error estándar

Al "ratio" o razón de expresiones se suele denominar también "Fold Change (FC)" porque en arrays de dos colores representaba cuantas veces más expresado está el gen en una (R) que en otra condición (G). Al logRatio también se le llama logFC y por extensión a la diferencia de medias en escala logarítmica tambien se la denomina logFC dado que se realiza de forma implícita la aproximación siguiente:

$$\bar{X}_1 - \bar{X}_2 = \log \bar{Y}_1 - \log \bar{Y}_2 \simeq \log(\bar{Y}_1) - \log(\bar{Y}_2) = \log\left(\frac{\bar{Y}_1}{\bar{Y}_2}\right)$$

Un primer ejemplo

Consideremos la tabla siguiente que representa una matriz de expresión simplificada que contiene las expresiones relativas (por ejemplo entre tejido tumoral y sano del mismo individuo) de 5 genes en 6 muestras.

Gen	R1	R2	R3	R4	R5	R6
A	2.50	2.70	2.50	2.80	3.20	2.00
B	0.01	0.05	-0.05	0.01	0.00	0.00
C	2.50	2.70	2.50	1.80	20.00	1.00
D	0.50	0.00	0.20	0.10	-0.30	0.30
E	0.10	0.11	0.10	0.10	0.11	0.09

Podemos calcular las medidas descritas para el caso de una muestra para decidir si un gen está expresado o no lo está. Se discutirá más adelante como precisar esto pero de momento nos quedaremos con la idea de que si la medida escogida es (cercana a) cero el gen no está diferencialmente expresado y si es mayor o menor que cero si que lo está. Nos referimos a “cero” porque estamos hablando de logaritmos de razones: si la expresión es la misma en ambas condiciones el cociente es uno y su logaritmo es cero.

Vale la pena insistir en el concepto de expresión diferencial: no nos preocupa cuáles es la expresión del gen en una u otra muestra sino si son distintas.

Gen	Promedio	Err. Std	T-test
A	2.617	0.397	14.735
B	0.003	0.032	0.233
C	5.083	7.335	1.550
D	0.133	0.273	1.091
E	0.102	0.008	30.200

La tabla anterior sugiere que podría considerarse el gen A está diferencialmente expresado (promedio y t-test altos) mientras que el gen B o el D no lo están (promedio y test-t próximos a cero). Los genes C y D pueden llevar a conclusiones contradictoria según nos basemos en el promedio o el test t.

Si se observan los valores del test t del gen C se concluye que el gen no aparece estar diferencialmente expresado. Si en cambio se observa su promedio parece que si que lo esté.

En el gen E pasa exactamente lo opuesto. La explicación de estas aparentes contradicciones se halla en el error estándar. En el gen C es muy elevado, debido a que el valor (20) es probablemente un “outlier”. En el gen D el error estándar es muy bajo por lo que, al encontrarse en el denominador del t-test aumenta artificialmente su valor.

7.1.2 Selección de genes diferencialmente expresados

Vamos a plantear la forma como se aborda este problema en el estudio de datos de microarrays. Dadas las características propias de este tipo de datos se consideran otras formas de estimar el error estándar que no sean tan sensibles a valores extremos o muy bajos.

Consideremos el gen g . Si llamamos:

- R_g log-ratio medio observado.
- SE_g error estándar de R_g estimado a partir de los datos en el gen g .
- SE error estándar de R_g estimado a partir de los datos con la información de todos los genes.

Podemos considerar dos variantes para el test- t :

- *Test-t global*: se calcula en base a un único estimador de SE para todos los genes:

$$t = R_g/SE,$$

- *Test-t específico*: Utiliza un estimador distinto del error estandar para cada gen:

$$t = R_g/SE_g.$$

Cada aproximación tiene sus pros y sus contras como muestra la tabla siguiente:

Test	Pros	Contras
Test-t Global	Estimador estable de σ	Asume homocedasticidad
Test-t específico	Robusto a heterocedasticidad	Estimador de σ no estable

En la práctica muchos métodos de selección de genes diferencialmente expresados han acabado buscando un compromiso entre ambas aproximaciones para lo que proponen o derivan fórmulas que de alguna forma ponderan o combinan dos estimaciones del error estándar: una basada en todos los genes y otra específica de cada gen. La tabla siguiente ilustra como algunos de los métodos más utilizados en la bibliografía incorporan esta idea.

Método (Referencia)	Fórmula
SAM (Tibshirani et al 2001)	$S = \frac{R_g}{c + SE_g}$
Cyber-T (Baldi et al, 2001)	$t = \frac{R_g}{\sqrt{\frac{v_0 SE^2 + (n-1) SE_g^2}{v_0 + n - 2}}}$
T-moderado (Smyth, 2003)	$t = \frac{R_g}{\sqrt{\frac{d_0 \cdot SE_0^2 + d \cdot SE_g^2}{d_0 + d}}}$

Al hecho de incluir un coeficiente que tenga en cuenta la variabilidad de todos los genes en el array para estimar el error estándar de cada gen se le denomina moderación de la varianza (“variance shrinkage”) y es una de las aproximaciones en que existe cierto consenso ([2]) acerca de que sirven para mejorar la selección de genes diferencialmente expresados.

Ejemplo de utilización del test-t

Como ejemplo utilizaremos el conjunto de datos `celltypes` y supondremos que disponemos ya de los datos normalizados y filtrados almacenados en un objeto `expressionSet`.

Si consideramos que el campo `treat` del objeto contiene la información de los grupos a comparar (ratones estimulados con LPS frente NA `rowttests` del paquete `genefilter` que realiza un test *t* sobre cada una de las filas (genes) de una matriz de expresión.

```
> stopifnot(require(BioBase))
> load ("celltypes-normalized.rma.Rda")
> my.eset <- eset_rma_filtered
> grupo_1 <- as.factor(pData(my.eset)$treat)
> stopifnot(require(genefilter))
> teststat <- rowttests(my.eset, "treat")
> print(teststat[1:5,])
```

	statistic	dm	p.value
1415694_at	3.503426	0.9813765	5.693821e-03
1415698_at	-4.253446	-0.9200460	1.680334e-03
1415743_at	-15.642231	-1.0156246	2.335398e-08
1415760_s_at	-5.686252	-0.8663532	2.020937e-04
1415772_at	11.394671	1.1306309	4.745989e-07

Cuanto mayor sea el valor absoluto del estadístico *t* mayor es la probabilidad de que el gen esté diferencialmente expresado.

Genes diferencialmente expresados estadísticamente significativos

Como hemos visto en la sección anterior dos medidas naturales para la selección de genes son el promedio de “log-ratios” –o la diferencia de promedios en el caso de muestras independientes– o el valor del estadístico de test (*t*-test) de una o dos muestras según si se trata de muestras apareadas o independientes respectivamente. Los primeros estudios de microarrays eran muy costosos y se hacían con pocas o incluso ninguna réplica por condición experimental. En estas situaciones la única forma fiable de detectar una diferencia de expresión era a través del “log-ratio” o sus diferencias.

Rápidamente se puso en evidencia que para poder obtener los genes que estaban realmente diferencialmente expresados era preciso disponer de un soporte estadístico que permitiera tener en cuenta la variación aleatoria existente entre muestras.

En la práctica esto se reduce a afirmar que si, además de la diferencia de expresión entre las condiciones experimentales, se lleva a cabo un test estadístico dispondremos de una medida objetiva, el *p*-valor que nos servirá para decidir qué genes se declaran diferencialmente expresados, a saber, aquellos en los que el *p*-valor del test sea inferior a un cierto umbral como 0.05 o 0.01.

Tal como se ha indicado en el capítulo 3, un test estadístico procede decidiendo rechazar la hipótesis nula si el *p*-valor es más pequeño que el nivel de significación del test.

Siguiendo con el ejemplo anterior podemos ordenar los resultados de los tests en base a los *p*-valores:

```
> ranked <- teststat[order(teststat$p.value),]  
> print(ranked[1:5,])
```

	statistic	dm	p.value
1449383_at	-48.61014	-2.044928	3.276616e-13
1430127_a_at	46.90904	1.508843	4.672074e-13
1451421_a_at	-40.72173	-1.445182	1.909283e-12
1450826_a_at	37.62239	5.147992	4.193941e-12
1416122_at	34.66314	1.482676	9.460684e-12

Ahora podríamos seleccionar, por ejemplo los genes cuyo *p*-valor fuera inferior a 0.01

```
> selectedTeststat <- ranked[ranked$p.value < 0.01,]
```

Esto deja un total de 1594 con un p–valor inferior a 0.01

“Volcano plots”

Si se opta por computar los valores de significación (p–valores) de los genes, resulta interesante comparar el tamaño del cambio del nivel de significación estadístico. El “volcano plot” es una representación gráfica que permite ordenar los genes a lo largo de dos dimensiones, la biológica, representada por el “fold change” y la estadística representada por el logaritmo negativo del p–valor.

En la escala horizontal se representa el cambio entre los dos grupos (en escala logarítmica, de manera que la regulación positiva o negativa se representa de forma simétrica). En la escala vertical se representa el p–valor del test en una escala logarítmica negativa, de forma que los p–valores más pequeños aparecen mayores.

Así pues puede considerarse que el primer eje indica impacto biológico del cambio (a más efecto biológico mayor “fold-change”) y el segundo muestra la evidencia estadística, o la fiabilidad del cambio.

La figura 43 muestra un “volcano–plot” para ejemplo de los “celltypes”.

Figura 43. Volcano plot

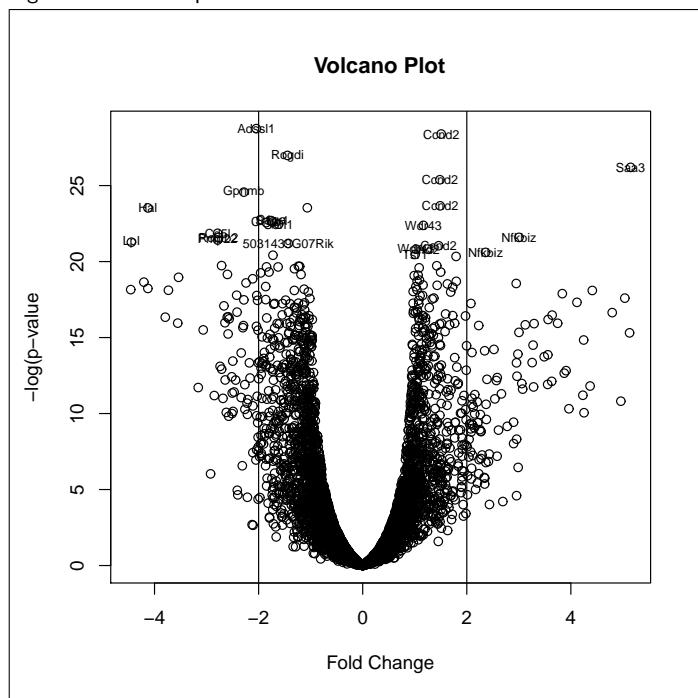


Figura 43

Ejemplo de Volcano plot que muestra los genes candidatos a considerarse como diferencialmente expresados en la comparación “LPS” frente a “Medium”

7.1.3 Potencia y tamaño muestral

Tal como se ha indicado más arriba, para realizar un *buen test* suele controlarse la probabilidad de error de tipo I (de falsos positivos) y buscar, de entre los tests candidatos, aquellos que tengan una menor probabilidad de error de tipo II, o equivalentemente una mayor potencia. A partir de este planteamiento existe, en el contexto estadístico estándar, multitud de formas de determinar cual debe ser el tamaño muestral necesario para obtener una potencia dada fijados el tamaño de efecto (“fold-change”) y la probabilidad de error de tipo I.

En el caso de los microarrays se han desarrollado diversas fórmulas para realizar cálculos de este tipo, pero la compleja estructura de los datos microarrays hace que sean relativamente *discutibles*.

Simon ([46]) sugiere la fórmula siguiente que es una generalización de las fórmulas clásicas para problemas de dos muestras:

El tamaño total requerido para detectar genes diferencialmente expresados en al menos una diferencia δ con una probabilidad de error de tipo I (FP), α y una probabilidad de error de tipo II (FN) $1 - \beta$ se calcula:

$$n = \frac{4(z_{\alpha/2} + z_{\beta})^2}{(\delta/\sigma)^2},$$

donde z_{α} y z_{β} son los percentiles $100\alpha/2$ y 100β de la distribución Normal $N(0, 1)$ y σ es la desviación estandar de un gen dentro de una clase (de un grupo). Obviamente σ es siempre desconocida por lo que, sin una muestra piloto con que estimarla el calculo es más imaginativo que realista.

Además de esto, el número de arrays usualmente recomendado queda lejos de la cantidad asequible para la mayor parte de los experimentos ([36, 49]). Lo que muchos usuarios hacen a la práctica, es buscar un equilibrio entre los costes y la reproducibilidad y, de hecho, tienden a usar una cantidad fija de arrays tal como 3 o 5 sin consideraciones adicionales.

Por ejemplo si ponemos $\alpha = 0.001$, $1 - \beta = 0.95$, $\delta = 1$ y estimamos σ entre todas las muestras el número de réplicas biológicas que necesitaremos será de 35.8.

7.1.4 El problema de la multiplicidad de tests (“multiple testing”)

El análisis de microarrays se realiza en base gen a gen e involucra múltiples tests, miles probablemente. Esto significa que, a medida que crece el número de genes, la probabilidad de declarar erroneamente al menos un gen diferencialmente expresado va en aumento, y si no se realiza algún tipo de ajuste el número de falsos positivos será tanto más alto cuantos más genes estemos analizando.

Hay muchas formas de intentar controlar estas probabilidades de error y puede verse un excelente revisión en Dudoit [15]).

De forma simplificada consideramos las dos aproximaciones más utilizadas.

Una posibilidad es mirar de controlar la probabilidad de obtener *al menos un falso positivo* o “Family-wise-error-rate (FWER)”. El más popular de estos métodos de control es la corrección de Bonferroni, consistente en multiplicar el p–valor por el número de tests realizados. La misma Dudoit y muchos otros autores han desarrollado variantes de los métodos clásicos de ajuste FWER usando por ejemplo tests de permutaciones.

El criterio FWER es quizás demasiado restrictivo dado que el control de los falsos positivos implica un considerable incremento de falsos negativos. En la práctica, sin embargo, muchos biólogos parecen estar dispuestos a aceptar que se produzcan algunos errores, siempre y cuando esto permita realizar descubrimientos. Por ejemplo un investigador debe considerar aceptable cierta pequeña proporción de errores (digamos del 10 al 20%) entre sus descubrimientos. En este caso, el investigador está expresando interés en controlar el porcentaje de falsos descubrimientos (FDR), es decir lo que es la proporción de falsos positivos sobre el total de genes inicialmente identificados como expresados diferencialmente. A diferencia del nivel de significación que queda determinado antes de examinar los datos, la FDR es una medida de confianza a posteriori ya que emplea información disponible en los datos para estimar las proporciones de falsos positivos que han tenido lugar. Si se obtiene una lista de los genes expresados diferencialmente en los que el FDR se controla hasta, digamos, el 20%, cabe esperar que el 20% de estos genes representen falsos positivos. Lo cual supone un enfoque menos restrictivo que controlar el FWER.

La decisión de controlar el FDR o el FWER depende de los objetivos del experimento. Si, por ejemplo, el objetivo es la *captura de genes* es razonable permitir cierta cantidad de falsos positivos y es preferible seleccionar FDR. Si por el contrario se trabaja con una lista de un tamaño menor al deseado para verificar la expresión de ciertos genes específicos, entonces el FWER es el criterio apropiado.

El ejemplo siguiente muestra como se realiza el ajuste de p–valores usando el paquete **multtest** de Bioconductor para ajustar por los métodos de Bonferroni (FWER), Benjamini & Hochberg (FDR) o Benjamini & Yekutieli (BY, FDR).

```
> stopifnot(require(multtest))
> procs <- c("Bonferroni", "BH", "BY")
> adjPvalues <- mt.rawp2adjp(teststat$p.value, procs)
> names(adjPvalues)

[1] "adjp"     "index"    "h0.ABH"   "h0.TSBH"
```

```
> ranked.adjusted<-cbind(ranked, adjPvalues$adjp)
> head(ranked.adjusted)

      statistic      dm   p.value     rawp Bonferroni
1449383_at -48.61014 -2.044928 3.276616e-13 3.276616e-13 1.478082e-09
1430127_a_at 46.90904  1.508843 4.672074e-13 4.672074e-13 2.107573e-09
1451421_a_at -40.72173 -1.445182 1.909283e-12 1.909283e-12 8.612774e-09
1450826_a_at 37.62239  5.147992 4.193941e-12 4.193941e-12 1.891887e-08
1416122_at    34.66314  1.482676 9.460684e-12 9.460684e-12 4.267714e-08
1448303_at    -31.92365 -2.283765 2.140612e-11 2.140612e-11 9.656299e-08

      BH        BY
1449383_at 1.053786e-09 9.475226e-09
1430127_a_at 1.053786e-09 9.475226e-09
1451421_a_at 2.870925e-09 2.581421e-08
1450826_a_at 4.729717e-09 4.252773e-08
1416122_at   8.535429e-09 7.674717e-08
1448303_at   1.609383e-08 1.447093e-07
```

Si seleccionamos los genes en base a su p–valor ajustado por ejemplo por el método de Benjamini y Yekutieli se obtienen los siguientes genes

```
> selectedAdjusted<-ranked.adjusted[ranked.adjusted$BY<0.001,]
> nrow(selectedAdjusted)
```

[1] 449

7.2 Modelos lineales para la selección de genes: limma

En la sección anterior se ha discutido el uso del test t y sus extensiones para la selección de genes diferencialmente expresados en situaciones relativamente sencillas, es decir cuando sólo hay dos grupos.

En muchos estudios el número de grupos a considerar es más de dos y las fuentes de variabilidad pueden ser más de una, por ejemplo una puede ser el tratamiento pero otra puede ser la edad de los individuos o cualquier otra condición fijada por el investigador o derivada de la heterogeneidad de las muestras.

En estas situaciones una aproximación razonable en problemas con *una variable respuesta* es el análisis de la varianza, discutido en el capítulo ???. En esta sección se presenta una aproximación equivalente que de forma general se denomina *el modelo lineal*. Este método –que engloba el análisis de la varianza y la regresión– es uno de los más usados en estadística y ha sido popularizado en el campo de los microarrays gracias a los trabajos de Gordon Smyth quien ha creado el paquete

`limma` que se ha convertido en la herramienta más utilizada para el análisis de microarrays.

7.2.1 El modelo lineal general

El modelo lineal (ver por ejemplo Faraway, [19]) es un marco general para la modelización y el análisis de datos estadística.

EL método consiste en asumir una relación lineal entre los valores observados de una variable *respuesta* y las condiciones experimentales. A partir de aquí se obtienen estimadores para los parámetros del modelo y de sus errores estándar, y (con algunas condiciones extra) es posible hacer inferencia acerca del experimento.

La aplicación de modelos lineales puede ser visto como un proceso secuencial, con los siguientes pasos:

- 1) Especificar el diseño del experimento: qué muestras se asignan a qué condiciones.
- 2) (Re-)Escribir un modelo lineal para este diseño en forma de $\mathbf{Y} = \mathbf{X}\beta + \varepsilon$, donde \mathbf{X} se denomina *la matriz de diseño*.
- 3) Una vez que el modelo se ha especificado aplicar la teoría general de estimar los parámetros y los contrastes (comparaciones entre los valores de los parámetros).
- 4) Si se cumplen ciertas condiciones de validez para el modelo es posible realizar inferencia sobre los parámetros del modelo, es decir se pueden contrastar hipótesis sobre dichos parámetros.

El esquema anterior se puede aplicar a casi cualquier tipo de situación experimental. En la sección siguiente se presentan un par de ellas.

7.2.2 Ejemplos de situaciones *modelizables* linealmente

Ejemplo 1: Experimento “Swirl–Zebrafish”

Swirl es una mutación puntual que provoca defectos en la organización del embrión en desarrollo a lo largo de su eje dorsal-ventral. Como resultado, algunos tipos celulares se reducen y otros se expanden. Un objetivo de este trabajo fue identificar los genes con expresión alterada en el mutante Swirl en comparación con ”wild zebrafish“.

- 1) El diseño experimental para este estudio fue el siguiente:

Slide	Cy3	Cy5
1	W	M
2	M	W
3	W	M
4	M	W

- Cada microarray contenía 8848 sondas de cDNA (genes o secuencias EST).
- Cuatro réplicas por array (slide): 2 juegos de pares de intercambio de color
- El cDNA del mutante swirl (S) se marca con Cy5 o Cy3 y el cDNA del "wild type" se marca con el otro

2) El modelo lineal derivado del diseño anterior fue:

- El parámetro de interés es: $\alpha = \mathbf{E}(\log \frac{S}{W})$.
- Las muestras 1 y 3 están marcadas con : S (Verde="Green") y W (Rojo="Red"), y las muestras 2 y 4 son "dye-swapped".
- El modelo, $\mathbf{y} = \mathbf{X}\alpha + \boldsymbol{\varepsilon}$, es:

$$\begin{aligned} y_1 &= \alpha + \varepsilon_1 \\ y_2 &= -\alpha + \varepsilon_2 \\ y_3 &= \alpha + \varepsilon_3 \\ y_4 &= -\alpha + \varepsilon_4 \end{aligned} \implies \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}}_{\text{Matriz de Diseño, } \mathbf{X}} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \alpha + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{pmatrix} \quad (9)$$

7.2.3 Ejemplo 2: Comparación de tres grupos

Los plásmidos IncHI codifican genes de resistencia múltiple a los antibióticos en *S. enterica*.

El plásmido R27 de la cepa salvaje es termosensible al transferirse.

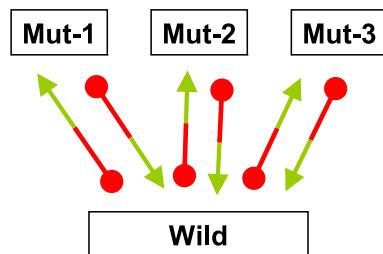
Algunos fenotipos mutantes relacionados con *hha* y *hns* cromosómicos participan en diferentes procesos metabólicos de interés en la conjugación termoregulada.

El objetivo del experimento es encontrar qué genes se expresan diferencialmente en tres tipos de mutantes diferentes, M_1 , M_2 y M_3 .

Posibles estrategias de diseño Este experimento debe ser planteado de forma diferente según el tipo de arrays (uno o dos colores) y qué comparaciones son las de mayor interés.

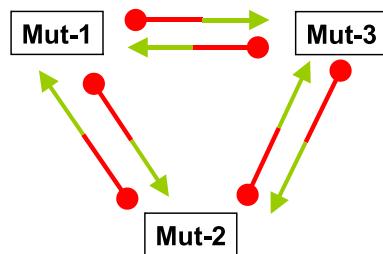
- Array de dos colores
- A *diseño de referencia*: Hibridar cada Mutante (M_i) vs. Salvaje (“Wild type”) (W).
- A *diseño loop*: Hibridar cada mutante el uno al otro en un doble “loop” que incluye (“dye-swapping”).
- Array de un color: hibridar mutantes y “wild types” separadamente.

Representación del diseño de referencia



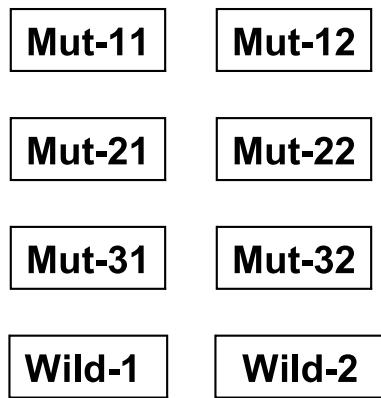
- Permite la comparación directa de Mutantes vs “Wild”.
- Número de parámetros a estimar es 3, relación intuitiva entre el número de parámetros y mutantes.
- Las comparaciones de Mutante vs Mutante son menos eficientes.

Representación del diseño de “loop” (bucle)



- Permite la comparación directa de Mutantes vs Mutantes.
- El número de parámetros a estimar es 2. Menos intuitivo.
- Las comparaciones Mutante vs Mutante son más eficientes.

Representación del diseño del array de un color



- Permite la comparación directa de
 - Mutante vs “Wild”
 - Mutante vs Mutante
- El número de parámetros a estimar es 4.
- Todas las comparaciones se pueden hacer de forma eficiente.

Modelo lineal para el diseño de referencia Modelo, $\mathbf{y} = \mathbf{X}\alpha + \varepsilon$, y contrastes $\mathbf{C}'\beta$

- Parámetros del modelo:

$$\alpha_1 = \mathbf{E} \left(\log \frac{M_1}{W} \right), \quad \alpha_2 = \mathbf{E} \left(\log \frac{M_2}{W} \right), \quad \alpha_3 = \mathbf{E} \left(\log \frac{M_3}{W} \right).$$

- *Contrastes:* Comparaciones interesantes.

$$\beta_1 = \alpha_1 - \alpha_2,$$

$$\beta_2 = \alpha_1 - \alpha_3,$$

$$\beta_3 = \alpha_2 - \alpha_3.$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}}_{\text{Matriz de Diseño, } \mathbf{X}} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix} \quad (10)$$

$$\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix}}_{\text{Matriz de Contraste, } \mathbf{C}} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}. \quad (11)$$

Modelo lineal para el diseño de “loop” Modelo, $\mathbf{y} = \mathbf{X}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}$, y contrastes $\mathbf{C}'\boldsymbol{\beta}$

- Parámetros del modelo:

$$\alpha_1 = \mathbf{E} \left(\log \frac{M_1}{M_2} \right), \quad \alpha_2 = \mathbf{E} \left(\log \frac{M_2}{M_3} \right).$$

$$\alpha_3 \text{ no es necesaria: } \log \left(\frac{M_1}{M_3} \right) = \log \left(\frac{M_1}{M_2} \right) - \log \left(\frac{M_2}{M_3} \right).$$

- *Contrastes:* Algunas de las comparaciones deseadas son precisamente los parámetros.

$$\beta_1 = \alpha_1,$$

$$\beta_2 = \alpha_2,$$

$$\beta_3 = \alpha_1 + \alpha_2.$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \\ -1 & 0 \\ 0 & -1 \\ -1 & 1 \end{pmatrix}}_{\text{Matriz de Diseño, } \mathbf{X}} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix} \quad (12)$$

$$\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & +1 \end{pmatrix}}_{\text{Matriz de Contraste, } \mathbf{C}} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}. \quad (13)$$

Modelo lineal para el diseño del array de un color Modelo, $\mathbf{y} = \mathbf{X}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}$, y contrastes $\mathbf{C}^1'\boldsymbol{\beta}$, $\mathbf{C}^2'\boldsymbol{\beta}$

- Parámetros del modelo:

$$\alpha_1 = \mathbf{E}(\log M_1), \alpha_2 = \mathbf{E}(\log M_2), \alpha_3 = \mathbf{E}(\log M_3), \alpha_4 = \mathbf{E}(\log W).$$

- *Contrastes*: Dos posibles conjuntos de comparaciones interesantes.

1) Comparación entre tipo de mutantes ($\mathbf{C}^1'\boldsymbol{\beta}$)

$$\beta_1^1 = \alpha_1 - \alpha_2,$$

$$\beta_2^1 = \alpha_3 - \alpha_2,$$

$$\beta_3^1 = \alpha_2 - \alpha_3.$$

2) Comparación entre cada mutante y el wild type ($\mathbf{C}^2' \beta$)

$$\beta_1^2 = \alpha_4 - \alpha_1,$$

$$\beta_2^2 = \alpha_3 - \alpha_1,$$

$$\beta_3^2 = \alpha_2 - \alpha_1.$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Matriz de Diseño, } \mathbf{X}} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{pmatrix} \quad (14)$$

$$\begin{pmatrix} \beta_1^1 \\ \beta_2^1 \\ \beta_3^1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \end{pmatrix}}_{\text{Matriz de Contraste, } \mathbf{C}^1} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix}. \quad (15)$$

$$\begin{pmatrix} \beta_1^2 \\ \beta_2^2 \\ \beta_3^2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}}_{\text{Matriz de Contraste, } \mathbf{C}^2} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix}. \quad (16)$$

Ejemplo 3: Estudio de la influencia de las citoquinas en ratones viejos

El objetivo de este experimento es estudiar el efecto de las citoquinas sobre la estimulación de una sustancia (LPS) y ver como esta relación se ve afectada por

la edad.

Se trata de un modelo de un factor (tratamiento, assignable por el individuo) y un bloque (edad, no assignable, prefijada en cada ratón). En la práctica el modelo equivale al del análisis de la varianza de dos factores.

1) El diseño experimental para este estudio fue el siguiente:

Array	Tratamiento	Edad
1	LPS	Viejo
2	LPS	Joven
3	Medio	Viejo
4	Medio	Joven
5	LPS	Viejo
6	LPS	Joven
7	Medio	Viejo
8	Medio	Joven
9	LPS	Viejo
10	LPS	Joven
11	Medio	Viejo
12	Medio	Joven

- Se utilizaron microarrays de un color (Affymetrix).
 - Cada condición se replicó tres veces.
 - Las preguntas específicas a responder:
 - ¿Cuál es el efecto del tratamiento en ratones viejos?
 - ¿Cuál es el efecto del tratamiento en ratones jóvenes?
 - ¿En qué genes el efecto es diferente?.
- 2) En este caso se pueden considerar distintos modelos lineales derivados del hecho de que este experimento admite diferentes parametrizaciones:
- Factores separados con 2 niveles cada uno para tratamiento (LPS/Med), Edad (Joven/Viejo) y su interacción:

$$Y_{ijk} = \underbrace{\alpha_i}_{\text{Trat}} + \underbrace{\beta_j}_{\text{Edad}} + \underbrace{\gamma_{ij}}_{\text{Interacción}} + \varepsilon_{ijk}, \quad i = 1, 2, j = 1, 2, k = 1, 2, 3$$

Esta primera parametrización parece natural pero es más complicado confiar en ella para responder las preguntas propuestas.

- Un factor combinado con 4 niveles
(*LPS.Aged, Med.Aged, LPS.Young, Med.Young*)

$$Y_{ij} = \alpha_i + \varepsilon_{ij}, \quad i = 1, \dots, 4, j = 1, 2, 3.$$

Esta parametrización parece más rígida pero se adapta mejor para responder a las preguntas planteadas.

Aquí se adopta la segunda parametrización.

- Parámetros del modelo:

$$\alpha_1 = \mathbf{E}(\log LPS.Aged), \quad \alpha_2 = \mathbf{E}(\log Med.Aged),$$

$$\alpha_3 = \mathbf{E}(\log LPS.Young), \quad \alpha_4 = \mathbf{E}(\log Med.Young).$$

- *Contrastes*: Preguntas que interesa responder:

$$\beta_1^1 = \alpha_3 - \alpha_1, \quad \text{Efecto del tratamiento en ratones viejos}$$

$$\beta_2^1 = \alpha_4 - \alpha_2, \quad \text{Efecto del tratamiento en ratones jóvenes}$$

$$\beta_3^1 = (\alpha_3 - \alpha_1) - (\alpha_2 - \alpha_4), \quad \text{Interacción: diferencia entre efectos}$$

- Modelo lineal:

Modelo, $\mathbf{y} = \mathbf{X}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}$, y contrastes $\mathbf{C}'\boldsymbol{\beta}$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Matriz de Diseño, } \mathbf{X}} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{pmatrix} \quad (17)$$

$$\begin{pmatrix} \beta_1^1 \\ \beta_2^1 \\ \beta_3^1 \end{pmatrix} = \underbrace{\begin{pmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix}}_{\text{Matriz de Contraste, } \mathbf{C}} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} \quad (18)$$

7.2.4 Estimación e inferencia con el modelo lineal

Una vez se ha expresado el experimento como un modelo lineal:

$$\mathbf{E}(\mathbf{y}_g) = \mathbf{X}\boldsymbol{\alpha}_g, \quad \text{var}(y_g) = W_g \sigma_g, \quad (19)$$

es posible usar *la teoría estándar del modelo lineal* (ver [19]) para obtener:

- Estimación de los parámetros: $\hat{\alpha}_g (\approx \alpha)$.
- Desviación estándar de las estimaciones: $\hat{\sigma}_g = s_g (\approx \sigma)$.
- Error estándar de las estimaciones: $\widehat{\text{var} \hat{\alpha}_g} = V_g s_g^2$.

Estas estimaciones son la base para realizar inferencia sobre α i.e. test $H_0 : \alpha = 0$?, basado en el hecho que:

$$t_{gj} = \frac{\alpha_{gj}}{s_g \sqrt{v_{gj}}} \sim \text{Distribución de Student.} \quad (20)$$

De forma análoga pueden derivarse fórmulas para $\alpha_1 - \alpha_2$, es decir para decidir acerca de las comparaciones.

Los procedimientos de estimación y de inferencia no dependen de qué parametrización se ha adoptado, a pesar de que distintas parametrizaciones puedan dar lugar a distintos valores numéricos..

Fortaleza y debilidades del modelo lineal

El enfoque del modelo lineal es flexible y potente:

- Se puede adaptar a situaciones diferentes y complejas.
- Siempre produce buenas estimaciones (“BLUE”).
- Si las suposiciones son ciertas proporciona una base para la inferencia.

Por otro lado hay que tener en cuenta que, si las suposiciones no se cumplen, entonces las conclusiones deben tomarse con precaución.

Y lo que es peor, aún siendo válidas las suposiciones del modelo los resultados pueden verse afectados por el tamaño de la muestra de forma que, en muestras pequeñas donde pueden haber variaciones más grandes es fácil que se obtengan valores t no significativos o, por el contrario, excesivamente significativos (si la variación es muy reducida).

La metodología desarrollada por Smyth ([47]) basada en los resultados de Lönsted & Speed ([48]) está dirigida a abordar cómo hacer frente a estas debilidades.

7.2.5 Modelos lineales para Microarrays

Smyth ([47]) considera el problema de la identificación de genes que se expresan diferencialmente en las condiciones especificadas en el diseño de experimentos de microarrays. Como hemos dicho repetidamente la variabilidad de los valores de expresión difiere entre genes, pero la naturaleza paralela de la inferencia en microarrays sugiere la posibilidad de usar la información *de todos los genes a la vez* para mejorar la estimación de los parámetros, lo que puede llevar a una inferencia más fiable.

Básicamente lo que propone Smyth ([47]) es una solución en tres pasos:

- Se plantea el problema como un modelo lineal con una componente bayesiana ya que se supone que los mismos parámetros a estimar son variables (no constantes) con distribuciones *prior* que se estimaran a partir de la información de todos los genes.
- A continuación se obtienen las estimaciones de los parámetros del modelo. La aproximación utilizada garantiza que estos estimadores tienen un comportamiento robusto incluso para pequeño número de arrays.
- Finalmente se calcula un “odd-ratio” que viene a ser la probabilidad de que un gen esté diferencialmente expresado frente a la de que no lo esté y se asocia este valor denominado estadístico B con un estadístico t moderado y su p–valor.

$$B = \log \frac{P[\text{Afectado}|M_{ij}]}{P[\text{No Afectado}|M_{ij}]}$$

gen=i ($i = 1\dots N$), réplica=j ($j = 1, \dots, n$).

El hecho de trabajar con logaritmos permite poner el punto de corte en el cero: A mayor valor positivo más probable es que el gen esté diferencialmente expresado. A mayor valor negativo, más probable es que no lo esté

7.2.6 Implementación y ejemplos

El paquete de *Bioconductor limma* al que hemos hecho referencia en los párrafos anteriores implementa el método de Smyth para la regularización de la varianza lo que lo ha convertido en muy popular entre los usuarios de microarrays

El código siguiente muestra como se crea la matriz del diseño y los contrastes para realizar el análisis mediante modelos lineales;

	Aged.LPS	Aged.MED	Young.LPS	Young.MED
Aged_LPS_80L.CEL	1	0	0	0
Aged_LPS_86L.CEL	1	0	0	0
Aged_LPS_88L.CEL	1	0	0	0
Aged_Medium_81m.CEL	0	1	0	0
Aged_Medium_82m.CEL	0	1	0	0
Aged_Medium_84m.CEL	0	1	0	0
Young_LPS_75L.CEL	0	0	1	0
Young_LPS_76L.CEL	0	0	1	0
Young_LPS_77L.CEL	0	0	1	0
Young_Medium_71m.CEL	0	0	0	1

```

Young_Medium_72m.CEL      0      0      0      1
Young_Medium_73m.CEL      0      0      0      1
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$lev
[1] "contr.treatment"

> require(limma)
> cont.matrix <- makeContrasts (
+     LPS.in.AGED=(Aged.LPS-Aged.MED),
+     LPS.in.YOUNG=(Young.LPS-Young.MED),
+     AGE=(Aged.MED-Young.MED),
+     levels=design)
> cont.matrix

```

Contrasts

Levels	LPS.in.AGED	LPS.in.YOUNG	AGE
Aged.LPS	1	0	0
Aged.MED	-1	0	1
Young.LPS	0	1	0
Young.MED	0	-1	-1

NA estimar el modelo, estimar los contrastes y realizar las pruebas de NA puede considerarse diferencialmente expresado.

La penúltima instrucción ejecuta el proceso de regularización de NA NA obtener estimaciones de error mejoradas.

NA Fold-change *t*-moderados o *p*-valores ajustados que se utilizan para ordenar los genes de mas a menos diferencialmente expresados.

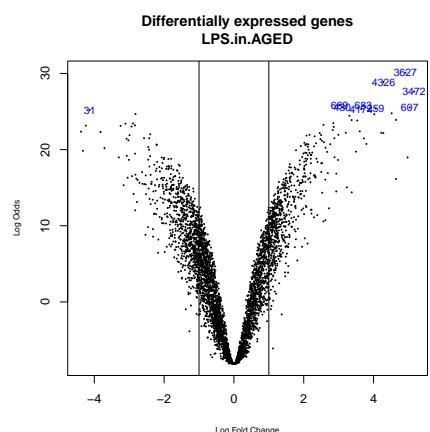
A fin de controlar el porcentaje de falsos positivos que puedan resultar del alto numero de contrastes realizados simultaneamente los p–valores se ajustan de forma que tengamos control sobre la tasa de falsos positivos utilizando el metodo de Benjamini y Hochberg ([4]).

La funcion **topTable** genera para cada contraste una lista de genes ordenados de mas a menos diferencialmente expresados.

```

> topTab_LPS.in.AGED <- topTable (fit.main, number=nrow(fit.main), coef="LPS.in.AGED", adjust="fdr")
> topTab_LPS.in.YOUNG <- topTable (fit.main, number=nrow(fit.main), coef="LPS.in.YOUNG", adjust="fdr")
> topTab_AGE <- topTable (fit.main, number=nrow(fit.main) , coef="AGE", adjust="fdr")

```



8. Después de la selección: Análisis de listas de genes

8.1 Introducción

El resultado de un análisis de microarrays como los descritos hasta el momento suele ser una “lista de genes”, es decir una archivo o una tabla con los identificadores de los genes considerados diferencialmente expresados en una o más comparaciones.

La pregunta obvia que los investigadores se plantean al disponer de estas listas es *?cuál es su significado biológico?*.

Para el bioinformático esa pregunta se replantea de la siguiente manera: ?Qué se puede hacer para convertir -de forma más o menos automática– la lista obtenida en algún tipo de información biológicamente relevante. Esta cuestión ha llevado al desarrollo de métodos y herramientas que de forma general se agrupan bajo el título de métodos para el *análisis de listas de genes* o métodos de *análisis de la significación biológica*.

Siendo este un campo muy extenso no lo vamos a desarrollar en profundidad sinó que nos limitaremos a considerar brevemente algunas aproximaciones sencillas que pueden contribuir a una mejor interpretación de los resultados de un experimento de microarrays.

Los aspectos que trataremos serán:

- Análisis comparativo de listas de genes.
- Anotación de los resultados.
- Visualización de la matriz de expresión para los genes seleccionados en una o más comparaciones.
- Análisis básico de la significación biológica: *Gen Enrichment Analysis*

8.2 Análisis comparativo de listas de genes

Muchos estudios analizan el comportamiento de los genes bajo varios tratamientos o condiciones experimentales.

En estos casos puede ser interesante ver como cambia un gen bajo distintos tratamientos o bien ver qué genes son afectados de forma parecida o distinta bajo los mismos. La forma habitual de hacer esta comparación es mediante algún programa que permita comparar los elementos de dos o más listas. En Bioconductor, por ejemplo el paquete **limma** tiene algunas funciones que permiten seleccionar los genes cambiados simultáneamente en dos o más condiciones.

8.2.1 Ejemplo: comparación de listas en el caso **celltypes**

Supondremos que tras las comparaciones del capítulo anterior disponemos de los valores de expresión y las listas de genes obtenidas de las tres comparaciones del caso **celltypes**. Ambos están almacenados en archivos binarios que podemos recuperar con la instrucción **load**.

```
> require(Biobase)
> require(limma)
> load("celltypes-normalized.rma.Rda")
> load("celltypes-fit.main.Rda")
> topTab_LPS.in.AGED <- topTable (fit.main, number=nrow(fit.main), coef="LPS.in.AGED", adjust="fdr", lfc=2)
> topTab_LPS.in.YOUNG <- topTable (fit.main, number=nrow(fit.main), coef="LPS.in.YOUNG", adjust="fdr", lfc=2)
> topTab_AGE <- topTable (fit.main, number=nrow(fit.main) , coef="AGE", adjust="fdr", lfc=2)
```

La función **decidetests** permite realizar ambas cosas. En este ejemplo no se ajustaran los p–valores entre comparaciones. Tan solo se seleccionaran los genes que cambian en una o más condiciones.

EL resultado del análisis es una tabla **res** que para cada gen y cada comparación contiene un 1 (si el gen esta sobre-expresado o “up” en esta condicion), un 0 (si no hay cambio significativo) o un -1 (si esta “down”-regulado).

Para resumir dicho análisis podemos contar qué filas tienen como mínimo una celda distinta de cero:

```
> sum.res.rows<-apply(abs(res),1,sum)
> res.selected<-res[sum.res.rows!=0,]
> print(summary(res))
```

	LPS.in.AGED	LPS.in.YOUNG	AGE
-1	787	841	921
0	3111	2980	2233
1	613	690	1357

```
> vennDiagram (res.selected[,1:3], main="Genes in common #1", cex=0.9)
```

Genes in common #1

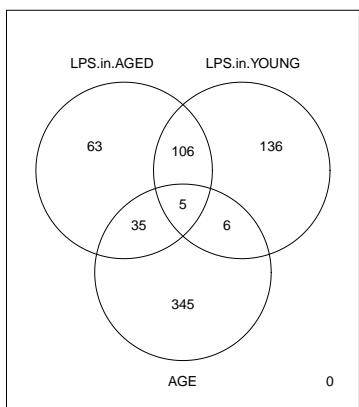


Figura 45: Número de genes seleccionado en cada comparación

En vista de estos valores podemos aplicar otros criterios de selección, por ejemplo genes con un p-valor ajustado inferior a 0.0001 y `log Fold change` mayor o igual a 2. *Este criterio combina a la vez la significación estadística y la significación biológica por lo que, en un estudio real sería probablemente el escogido.*

	LPS.in.AGED	LPS.in.YOUNG	AGE
-1	98	117	65
0	4302	4258	4120
1	111	136	326

Un diagrama de Venn permite visualizar la tabla anterior sin diferenciar entre genes “up” o “down” regulados.

8.3 Anotación de los resultados

La identificación de los genes seleccionados puede resultar más sencilla para el especialista en un campo si se utilizan nombres estándar como el símbolo del gen o “gene symbol”. Ahora bien hemos de recordar que los microarrays suele ser un producto comercial en los que la empresa que los crea decide qué sondas de cada gen o exón coloca en cada posición del array. Esto significa que para poder indicar el nombre de cada gen seleccionado es preciso disponer de algún tipo de *tabla de anotaciones* que relacione cada sonda con los genes a los que apunta. Esta tabla debería ser proporcionada por las compañías que producen los arrays pero es algo que queda a su criterio y, en el mejor de los casos, cada empresa puede seguir unos criterios distintos.

Afortunadamente si se trabaja con Bioconductor existe una solución homogénea a este problema. Se trata de los *paquetes de anotaciones* creados y mantenidos por el equipo de Bioconductor a partir de la información publicada por cada compañía pero con una forma común de acceso. Es decir aunque las distintas empresas utilicen formatos distintos desde el punto de vista del usuario de Bioconductor esto no importará: si existe el paquete se utiliza como todos los demás.

De hecho Bioconductor incorpora varios tipos de paquetes de anotaciones:

- **Paquetes de anotaciones de microarrays:** Asocian a cada sonda la información disponible para ella en distintas bases de datos. Consisten en una serie de tablas SQL (de hecho cada paquete es una base de datos SQL) que asocian las sondas de cada gen con sus correspondencias en distintas bases de datos.
- **Paquetes de anotaciones de organismos:** Contienen asociaciones similares a las anteriores pero las claves principales de cada tabla no son las sondas de un microarray determinado sino los genes de un organismo dado. Es decir en estos paquetes cada tabla asocia el identificador *entrez* del gen con sus correspondientes identificadores en otras bases de datos.
- **Paquetes de anotaciones de bases de datos** Estos paquetes difieren de los anteriores porque lo que hacen es almacenar en el mismo formato que los anteriores –bases de datos SQL consultables en R– una copia de ciertas bases de datos como la *Gene Ontology (GO)* o la *Kyoto Encyclopedia of Genes and Genomes (KEGG)*.

Para saber que anotaciones están disponibles debe cargarse el paquete y llamar la función del mismo nombre. Por ejemplo para los microarrays del caso `celltypes`:

Para saber que anotaciones están disponibles debe cargarse el paquete y llamar la función del mismo nombre sin el sufijo “.db”.

```
> if (!require(mouse4302.db)){  
+     biocLite("mouse4302.db")  
+ }  
> require(mouse4302.db)  
> anotData <- capture.output(mouse4302())  
> print(anotData[1:15])  
  
[1] "Quality control information for mouse4302:  
[2] ""  
[3] ""  
[4] "This package has the following mappings:  
[5] ""
```

```
[6] "mouse4302ACCNUM has 45101 mapped keys (of 45101 keys)"
[7] "mouse4302ALIAS2PROBE has 70701 mapped keys (of 128120 keys)"
[8] "mouse4302CHR has 38464 mapped keys (of 45101 keys)"
[9] "mouse4302CHRENGTHS has 35 mapped keys (of 35 keys)"
[10] "mouse4302CHRL0C has 34026 mapped keys (of 45101 keys)"
[11] "mouse4302CHRL0CEND has 34026 mapped keys (of 45101 keys)"
[12] "mouse4302ENSEMBL has 33575 mapped keys (of 45101 keys)"
[13] "mouse4302ENSEMBL2PROBE has 17074 mapped keys (of 20989 keys)"
[14] "mouse4302ENTREZID has 38535 mapped keys (of 45101 keys)"
[15] "mouse4302ENZYME has 4534 mapped keys (of 45101 keys)"
```

```
> cat ("... output continues until ", length(anotData), " lines.\n")
```

... output continues until 46 lines.

Si en vez del paquetes del microarray usáramos el paquete de organismo, en este caso del ratón.

```
> if (!(require(org.Mm.eg.db))){
+     biocLite("org.Mm.eg.db")
+ }
> require(org.Mm.eg.db)
> anotData <- capture.output(org.Mm.eg())
> print(anotData[1:15])
```

```
[1] "Quality control information for org.Mm.eg:"
[2] ""
[3] ""
[4] "This package has the following mappings:"
[5] ""
[6] "org.Mm.egACCNUM has 36333 mapped keys (of 57955 keys)"
[7] "org.Mm.egACCNUM2EG has 472125 mapped keys (of 472125 keys)"
[8] "org.Mm.egALIAS2EG has 128120 mapped keys (of 128120 keys)"
[9] "org.Mm.egCHR has 56836 mapped keys (of 57955 keys)"
[10] "org.Mm.egCHRENGTHS has 35 mapped keys (of 35 keys)"
[11] "org.Mm.egCHRL0C has 21224 mapped keys (of 57955 keys)"
[12] "org.Mm.egCHRL0CEND has 21224 mapped keys (of 57955 keys)"
[13] "org.Mm.egENSEMBL has 21575 mapped keys (of 57955 keys)"
[14] "org.Mm.egENSEMBL2EG has 20989 mapped keys (of 20989 keys)"
[15] "org.Mm.egENSEMBLPROT has 21553 mapped keys (of 57955 keys)"
```

```
> cat ("... output continues until ", length(anotData), " lines.\n")
```

```
... output continues until 55 lines.
```

Cada tabla de asociación puede consultarse de diversas formas,

- Con las funciones `get` o `mget`.
- Convirtiéndola en una tabla y extrayendo valores
- En algunos casos utilizando funciones específicas como `getSYMBOL` o `getEG` (por “Entrez Gene”) cuando existan.

Por ejemplo si tomamos los cinco primeros genes seleccionados en la comparación “LPS.in.AGED”

```
> top5 <- topTab_LPS.in.AGED$ID[1:5]
> cat("Usando mget\n")
```

Usando `mget`

```
> geneSymbol5.1 <- unlist(mget(top5, mouse4302SYMBOL))
> geneSymbol5.1
```

```
1450826_a_at 1457644_s_at 1449450_at 1419607_at 1419681_a_at
"Saa3" "Cxcl1" "Ptges" "Tnf" "Prok2"
```

```
> cat("Usando toTable\n")
```

Usando `toTable`

```
> genesTable<- toTable(mouse4302SYMBOL)
> rownames(genesTable) <- genesTable$probe_id
> genesTable[top5, 2]
```

```
[1] "Saa3" "Cxcl1" "Ptges" "Tnf" "Prok2"
```

```
> cat("Usando getSYMBOL\n")
```

Usando `getSYMBOL`

```
> require(annotate)
> geneSymbol5.3 <- getSYMBOL(top5, "mouse4302.db")
> geneSymbol5.3

1450826_a_at 1457644_s_at 1449450_at 1419607_at 1419681_a_at
  "Saa3"      "Cxcl1"     "Ptges"      "Tnf"       "Prok2"
```

Bioconductor dispone de algunos paquetes que permiten aprovechar esta funcionalidad anterior para obtener las anotaciones de cada gen y generar una tabla HTML con enlaces a algunas bases de datos.

De forma sencilla es posible obtener tablas con las anotaciones correspondientes a los genes seleccionados. Si se desea ser más ambicioso es posible generar tablas en las que se combinen hiperenlaces a las anotaciones con los resultados de la selección de genes.

El paquete **annaffy** permite de forma muy simple generar una tabla de anotaciones con hiperenlaces a las bases de datos para cada anotación seleccionada.

La instrucción siguiente crearía una tabla con las anotaciones disponibles para los genes seleccionados en la sección de comparaciones múltiples.

```
> require(annaffy)
> genesSelected <- rownames(res.selected)
> at <- aafTableAnn(genesSelected, "mouse4302.db")
> saveHTML (at, file.path(resultsDir, "anotations.html"),
+           "Annotations for selected genes")
```

8.4 Visualización de los perfiles de expresión

Tras seleccionar los genes diferencialmente expresados podemos visualizar las expresiones de cada gen agrupándolas para destacar los genes que se encuentran up o down regulados simultáneamente constituyendo *perfiles de expresión*.

Hay distintas formas de visualización pero aquí tan sólo se presenta el uso de mapas de color o **Heatmaps** cuyos fundamentos se explican en el capítulo dedicado al descubrimiento de clases.

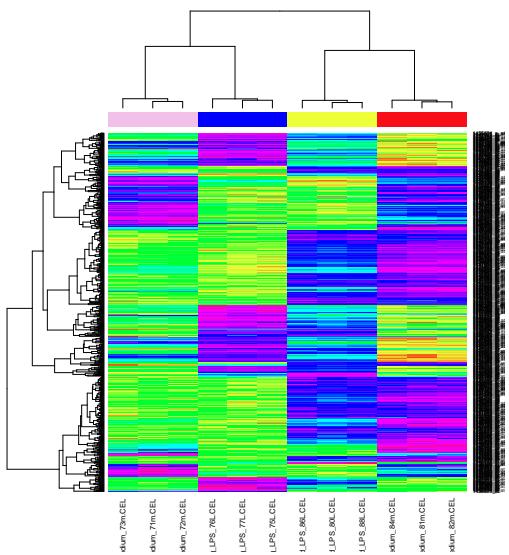
En primer lugar seleccionamos los genes a visualizar: Se toman todos aquellos que han resultado diferencialmente expresados en alguna de las tres comparaciones.

```
> probeNames<-rownames(res)
```

```
> probeNames.selected<-probeNames[sum.res.rows!=0]
> exprs2cluster <-exprs(eset_rma_filtered)[probeNames.selected,]
```

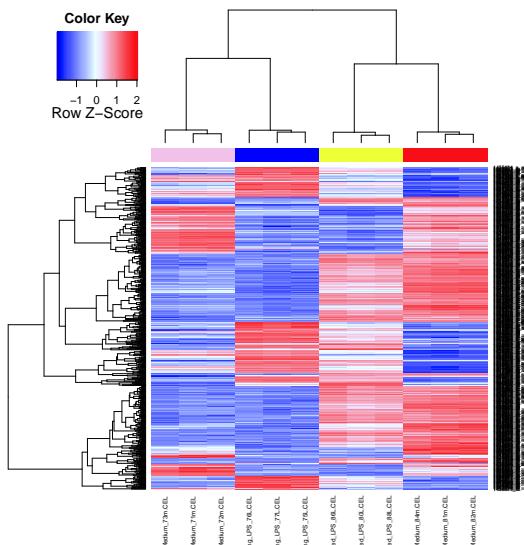
Para representar el Heatmap tan sólo necesitamos la matriz de datos resultante.

```
> color.map <- function(grupo) {
+   switch(grupo,
+         "yellow",
+         "red",
+         "blue",
+         "pink")
+ }
> grupColors <- unlist(lapply(pData(eset_rma_filtered)$grupo, color.map))
> heatmap(exprs2cluster, col=rainbow(100), ColSideColors=grupColors, cexCol=0.7)
```



Si se desea realizar mapas de color más sofisticados puede utilizarse el paquete *gplots* que implementa una versión mejorada en la función **heatmap.2**

```
> require("gplots")
> heatmap.2(exprs2cluster,
+             col=bluered(75), scale="row",
+             ColSideColors=grupColors, key=TRUE, symkey=FALSE,
+             density.info="none", trace="none", cexCol=0.6)
```



8.5 Análisis de significación biológica

En las secciones anteriores se ha visto como encontrar los identificadores de los genes en distintas bases de datos, lo que permite por ejemplo conocer sus nombres comunes (“gene symbol”).

Otra aproximación razonable es estudiar las funciones de los genes buscando sus anotaciones en bases de datos de *anotación funcional* como la *Gene Ontology* (GO), o la *Kyoto Encyclopedia of Genes and Genomes*.

Por ejemplo las anotaciones en GO para los cinco primeros genes de la lista analizada en ?? serían:

```
> require(annotate)
> top1 <- topTab_LPS.in.AGED$ID[1:1]
> geneSymbol1 <- getSYMBOL(top1, "mouse4302.db")
> GOAnots1 <- mget(top1, mouse4302GO)
> for (i in 1:length(GOAnots1)){
+   for (j in 1:length(GOAnots1[[i]])){
+     GOAnot <- GOAnots1[[i]][[j]][[1]]
+     cat(top1[i], geneSymbol1[i], GOAnot, substr(Term(GOAnot), 1, 30), "\n")
+   }
+ }
```

```
1450826_a_at Saa3 GO:0006953 acute-phase response
1450826_a_at Saa3 GO:0005576 extracellular region
1450826_a_at Saa3 GO:0034364 high-density lipoprotein parti
```

Como se ve en el ejemplo el número de anotaciones para un gen en la Gene

Ontology es muy alto, aparte de que, aunque aquí no se muestra, no todas las anotaciones tienen la misma fiabilidad.

A parte del problema de lo extenso de las anotaciones está el hecho de que antes de empezar a hacer inferencias sobre el significado de una anotación debería poderse establecer si dicha anotación está relacionada con el proceso que se está estudiando o aparece por azar entre la muchas anotaciones de los genes de la lista. Hay diferentes métodos y modelos para hacer esto (ver Draghici y colegas, [32] o Mosquera and Sánchez-Pla, [40, 43]) pero aquí se presentará brevemente lo que se conoce por *Análisis de enriquecimiento*.

Análisis de enriquecimiento

El objetivo del análisis de enriquecimiento es establecer si una determinada categoría, que representa, por ejemplo, un proceso biológico (GO) o una vía (KEGG), aparece más (“enriquecido”) o menos (“pobre”) a menudo en la lista de genes seleccionados que en la población (génica), desde donde se han obtenido, es decir, el array, el genoma, o simplemente los genes que fueron seleccionados para la prueba. La idea básica es que si una función aparece más a menudo en la lista que en general es probable que tenga algo que ver con el proceso en base al cual se ha seleccionado la lista que se estudia.

Por ejemplo, consideremos un experimento que da una lista de genes y el 10% de los genes más diferencialmente expresados están asociados con el término apoptosis en la GO (GO:0006915). Esto puede parecer una proporción inusualmente grande de la lista de genes, dado que la apoptosis es un proceso biológico muy específico. Para determinar cuánto más grande de lo normal es esta proporción, debe ser comparada con la proporción de genes relacionados con apoptosis en la lista de genes de referencia, que suele ser el conjunto de todos los genes del microarray.

El análisis estadístico realizado para comparar proporciones es un test Hipergeométrico o el test exacto de Fisher que se utiliza para probar la hipótesis:

$$H_0 : p_{sel}^A = p_{all}^A \text{ vs } H_1 : p_{sel}^A \neq p_{all}^A, \quad (21)$$

donde A representa el conjunto de genes cuya más/menos representación está siendo considerada, p_{sel}^A es la proporción de genes seleccionados que están incluidos en este conjunto de genes y p_{all}^A es la proporción de genes de la lista de referencia.

Análisis de enriquecimiento con Bioconductor

Hay muchas herramientas que pueden ayudar a realizar este análisis. Siguiendo nuestra costumbre usaremos las que contiene el paquete de R(Bioconductor), principalmente en el paquete **G0stats**.

El análisis realizado utilizando este paquete procede de la forma siguiente:

- Toma como entrada los identificadores de *Entrez* o de *Affy* de la lista de genes seleccionada así como el nombre del paquete de la anotación correspondiente al array que ha sido usado para el análisis.
- La salida del análisis es la lista de categorías que aparece más/menos representado en cada conjunto seleccionado.

El código siguiente ilustra como se procederá para realizar un análisis de enriquecimiento con las listas de genes seleccionados en la primera de las comparaciones realizadas en este capítulo.

```
> require(G0stats)
> require(mouse4302.db)
> if(!(require(org.Mm.eg.db)))
+   biocLite("org.Mm.eg.db")
> require(org.Mm.eg.db)
> # Seleccionamos la "topTable"
> topTab <- topTab_LPS.in.AGED
> # Definimos el universo de genes: todos los que se han incluido en el análisis
> # El programa trabaja con identificadores "entrez" y no admite duplicados
>
> entrezUniverse = unique(as.character(topTab$ID), "mouse4302.db"))
> # Escogemos los grupos de sondas a incluir en el análisis
> # Este análisis trabaja bien con varios centenares de genes
> # por lo que es habitual basarse en p-valores sin ajustar para incluirlos
>
> whichGenes<-topTab["adj.P.Val"]<0.001
> geneIds <- unique(as.character(topTab$ID[whichGenes]), "mouse4302.db"))
> # Creamos los "hiperparámetros" en que se basa el análisis
> G0params = new("GOHyperGParams",
+   geneIds=geneIds, universeGeneIds=entrezUniverse,
+   annotation="org.Mm.eg.db", ontology="BP",
+   pvalueCutoff=0.001, conditional=FALSE,
+   testDirection="over")
> KEGGparams = new("KEGGHyperGParams",
+   geneIds=geneIds, universeGeneIds=entrezUniverse,
+   annotation="org.Mm.eg.db",
+   pvalueCutoff=0.01, testDirection="over")
```

```
> # Ejecutamos los análisis
>
> GOhyper = hyperGTest(GOparams)
> KEGGhyper = hyperGTest(KEGGparams)
> # Creamos un informe html con los resultados
> comparison = "topTab_LPS.in.AGED"
> GOfilename = file.path(resultsDir,
+   paste("GOResults.",comparison,".html", sep=""))
> KEGGfilename = file.path(resultsDir,
+   paste("KEGGResults.",comparison,".html", sep=""))
> htmlReport(GOhyper, file = GOfilename, summary.args=list("htmlLinks"=TRUE))
> htmlReport(KEGGhyper, file = KEGGfilename, summary.args=list("htmlLinks"=TRUE))
```

9. Caso resuelto: Selección de genes diferencialmente expresados

9.1 Introducción

El análisis de datos de microarrays suele proceder, en general secuencialmente, a través de una serie de etapas tal y como se muestra en la figura 47.

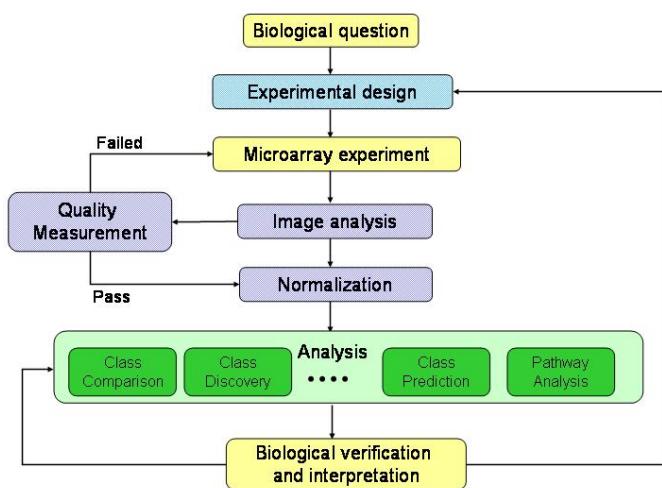


Figura 47: The Microarray Analysis Process

En cada una de las etapas pueden llevarse a cabo diversos procesos, con diversas variantes de métodos parecidos. A lo largo de la primera década del 2000 se ha desarrollado el proyecto Bioconductor que ha impulsado la creación de cientos de paquetes de R que implementan casi todas las capacidades posibles de análisis de microarrays. Aunque nadie puede conocer todos los paquetes con tan sólo conocer algunos de ellos es posible llevar a cabo potentes análisis con relativa facilidad.

El objetivo de este capítulo es ilustrar de forma breve pero completa como se hace un análisis de microarrays usando las facilidades que ofrecen Bioconductor y R.

Como se ha dicho, uno de los problemas en este tipo de estudios es que en cada etapa se puede proceder de varias formas lo que da lugar a un asfixiante número de posibilidades especialmente para el neófito. Con el fin de evitar este problema de momento se describe una sola de las opciones posibles para cada paso, lo que constituye un proceso “al estilo Bioconductor”

9.1.1 El ejemplo

El ejemplo analizado consiste en unos datos disponibles en forma de paquete (**estrogen**) en Bioconductor. Se trata de un estudio en el que se estudia la influencia del tratamiento con estrógeno y del tiempo transcurrido desde el tratamiento en la expresión de genes asociados con cancer de mama. En capítulos anteriores se han utilizado fragmentos de este ejemplo descrito en el capítulo 4 en el epígrafe correspondiente a los ejemplos (ejemplo 4.3.3).

El diseño experimental para este estudio consiste en un diseño factorial de 2 factores “Estrogeno” (Presente o Ausente), “Tiempo” (10h o 48h). La documentación del paquete **estrogen** ofrece más detalles acerca del diseño y las variables utilizadas. Si el paquete no se encuentra instalado puede instalarse con la instrucción **biocLite** que se descarga de la web de Bioconductor.

```
> if (!require("estrogen", character.only=T)){  
+   source("http://bioconductor.org/biocLite.R")  
+   biocLite("estrogen")  
+ }
```

En lo que sigue supondremos que se ha llevado a cabo una instalación estándar de Bioconductor es decir se han ejecutado las instrucciones:

```
> source("http://bioconductor.org/biocLite.R")  
> biocLite("estrogen")
```

9.1.2 Directorios y opciones de trabajo

Para facilitar supondremos que trabajamos en un directorio escogido por nosotros y cuya localización se asigna a la variable **workingDir**. Los datos se copiarán en un subdirectorio del anterior denominado “data” que se almacenará en la variable **dataDir** y los resultados se almacenarán en un directorio “results” cuyo nombre completo se almacenará en la variable **resultsDir**.

```
> workingDir <- getwd()  
> system("mkdir data")  
> system("mkdir results")  
> dataDir <- file.path(workingDir, "data")  
> resultsDir <- file.path(workingDir, "results")  
> setwd(workingDir)  
  
> options(width=80)  
> options(digits=5)
```

9.2 Obtención y lectura de los datos

Para un análisis de datos de microarrays de Affymetrix se necesitan los archivos de imágenes escaneadas (“.CEL”) y un archivo en el que se asigne una condición experimental a cada archivo.

9.2.1 Los datos

Una ventaja de este ejemplo es que los datos se encuentran disponibles tras instalar el paquete `estrogen` por lo que se pueden copiar un directorio de trabajo o simplemente trabajar con ellos en el directorio original. El directorio en que se encuentran los archivos .CEL es:

```
> require(estrogen)
> estrogenDir <- system.file("extdata", package = "estrogen")
> print(estrogenDir)

[1] "/home/alex/R/x86_64-pc-linux-gnu-library/2.15/estrogen/extdata"
```

Para realizar el análisis es preciso copiar todos los archivos del directorio “extdata” a nuestro directorio de trabajo “data”.

ATENCIÓN: Esta operación que depende de la instalación y del sistema operativo. Por ejemplo para copiarlos desde R en linx se escribirá:

```
> system(paste ("cp ", estrogenDir,"/* ./data", sep=""))
```

Los archivos copiados son

- Los ocho archivos .CEL para el análisis
- Un archivo defectuoso para que se vea como debe salir un “array malo” en el control de calidad.
- Un archivo de covariables o “targets” que se usará al leer los archivos .CEL incorporar la información de las covariables en el objeto que se creará.

9.2.2 Lectura de los datos

El proceso de leer los datos puede parecer un poco extraño a primera vista pero la idea es simple:

- En primer lugar creamos algunas estructuras de datos que contienen la información sobre las variables y - opcionalmente - sobre las anotaciones y el experimento y
- A continuación nos basamos en estas estructuras para leer los datos y crear los objetos principales que se utilizarán para el análisis.

```
> require(BioBase)
> require(affy)
> sampleInfo <- read.AnnotatedDataFrame(file.path(estrogenDir, "targLimma.txt"),
+   header = TRUE, row.names = 1, sep="\t")
> fileNames <- pData(sampleInfo)$FileName
> rawData <- read.affybatch(filenames=file.path(estrogenDir,fileNames),
+   phenoData=sampleInfo)
```

Una forma alternativa de leer los datos consiste en acceder al directorio en donde se encuentran y escribir `ReadAffy`. En este ejemplo se incluye un archivo defectuoso con fines didácticos, llamado “badcel”. Para ilustrar las diferencias entre archivos correctos e incorrectos se puede leer en otro objeto.

```
> require(affy)
> setwd(estrogenDir)
> rawData.wrong <- ReadAffy()
> setwd(workingDir)
```

9.3 Exploración, Control de Calidad y Normalización

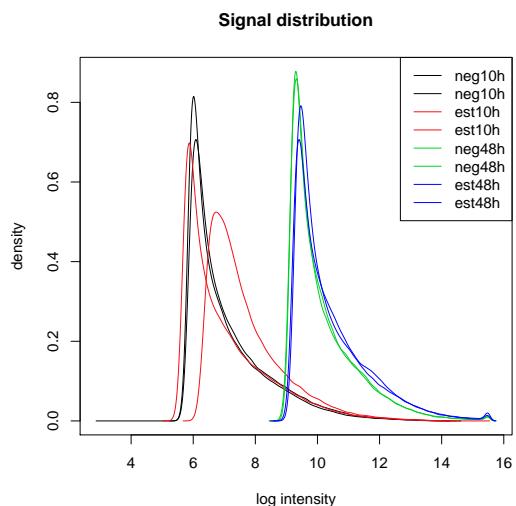
Tras leer los datos pasamos al preprocessado. Aunque puede interpretarse de distintas formas esta fase suele consistir en

- 1) Realizar algunos gráficos con los datos para hacerse una idea de como ha resultado el experimento.
- 2) Realizar un control de calidad más formal.
- 3) Normalizar y, en el caso de Affymetrix, resumir las expresiones

9.3.1 Exploración y visualización

La exploración previa puede hacerse paso a paso o en bloque si se utiliza algún paquete como `arrayQualityMetrics`. En la ayuda de este paquete se encuentra una descripción de los análisis básicos que permite realizar.

El primer gráfico que suele hacerse es algún tipo de “densidad” que muestre la distribución de las señales en los datos “crudos”, es decir sin normalizar. Estos gráficos también permiten hacerse una idea de si las distribuciones de los distintos arrays son similares en forma y posición.



El gráfico de degradación –que no aparece en este caso, ya que daba problemas al representarlo– permite hacerse una idea de como ha sido el proceso de hibridación de las muestras. Lineas paralelas sugieren una calidad similar.

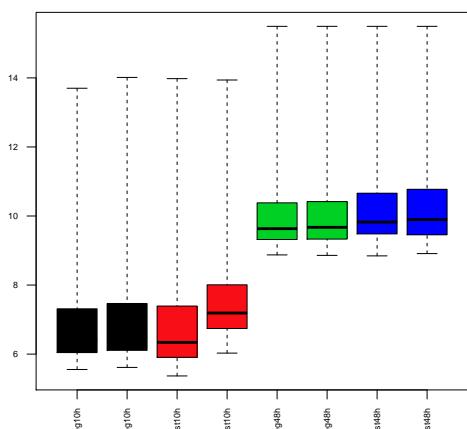
```

low10A low10B hi10A   hi10B   low48A   low48B   hi48A   hi48B
slope  -0.103 -0.217 -0.129 -0.3810 -0.50400 -0.55300 -0.3510 -0.66700
pvalue 0.550  0.194  0.394  0.0341  0.00482  0.00109  0.0464  0.00032

```

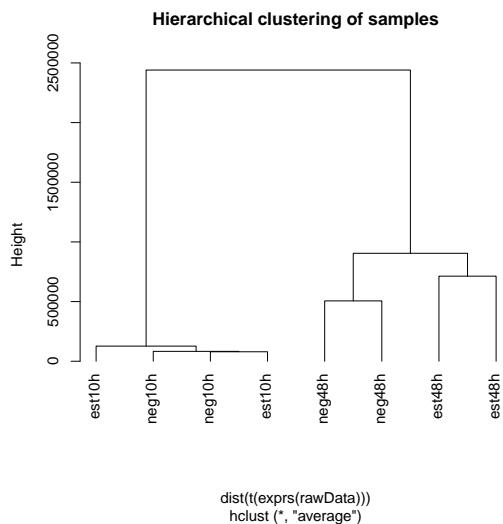
EL código para el gráfico de degradación sería:

El diagrama de cajas muestra, como el histograma, una idea de la distribución de los datos.



Finalmente un cluster jerárquico seguido de un dendrograma nos puede ayudar a hacernos una idea de si las muestras se agrupan por condiciones experimentales.

Si lo hacen es bueno, pero si no, no es necesariamente indicador de problemas, puesto que es un gráfico basado en todo los datos.



9.3.2 Control de calidad

Las exploraciones anteriores nos proporcionan una idea de como son los datos.

Se pueden realizar controles de calidad más estrictos como:

- Los controles de calidad estándar de Affymetrix, descritos en el paquete **simpleaffy**.
- Controles basados en modelos a nivel de sondas, descritos en el paquete **affyPLM**.

El paquete **affyQCReport** encapsula los análisis que pueden realizarse con el paquete **simpleaffy**, de forma que con una instrucción se pueden realizar todos los análisis y enviar la salida a un archivo.

```

> stopifnot(require(affyQCReport))
> QCReport(rawData,file=file.path(resultsDir,"QCReport.pdf"))
  
```

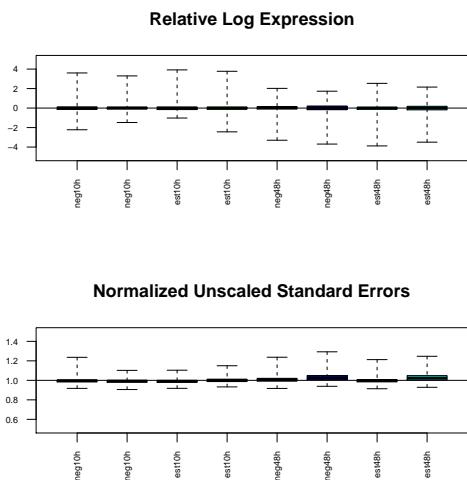
EL paquete **affyPLM** realiza un control de calidad basado en “probe-level models” (PLM).

```

> stopifnot(require(affyPLM))
  
```

```
> Pset<- fitPLM(rawData)
```

Como resultado del ajuste PLM se pueden obtener dos graficos, uno de expresiones relativas y otro con errores estandarizados (ver figura ??) Si los datos son de calidad ambos graficos deben ser centrados y relativamente simetricos. Cambios en esta situacion sugieren problemas en los arrays que no las verifiquen.



Para concluir con esta sección merece la pena tener en cuenta que todos estos graficos son exploratorios. Raramente se descarta un array si solo un grafico lo sugiere.

Merece la pena repetir la exploración y el control de calidad con el objeto `rawData.wrong`. En este caso hay un array defectuoso y se ve muy claramente que quiere decir “array problemático” en los graficos.

9.3.3 Normalización y Filtraje

Una vez realizado el control de calidad se procede a normalizar los datos y sumarizarlos.

Hecho esto puede realizarse un filtraje no específico con el fin de eliminar genes que constituyen básicamente “ruído”, bien porque sus señales son muy bajas o bien porque apenas varían entre condiciones, por lo que no aportan nada a la selección de genes diferencialmente expresados.

Normalización

La normalización puede hacerse por distintos métodos (MAS5, VSN, RMA, GCRMA, ...) En este ejemplo se utilizará el método RMA pero no se realizará

filtraje alguno. Esto puede implicar quizás que para seleccionar genes diferencialmente expresados basándose en el ajuste de p-valores debamos utilizar criterios menos restrictivos que si hubiéramos filtrado, pero tiene la ventaja de eliminar un paso que, en el mejor de los casos, resulta controvertido.

El procesado mediante RMA implica un proceso en tres etapas:

- Corrección de fondo (el RMA hace precisamente esto).
- Normalización para hacer los valores de los arrays comparables.
- Summarización de las diversas sondas asociadas a cada grupo de sondas para dar un único valor.

```
> stopifnot(require(affy))
> normalize <- T
> if(normalize){
+   eset_rma <- rma(rawData)
+   save(eset_rma, file=file.path(dataDir,"estrogen-normalized.Rda"))
+ }else{
+   load (file=file.path(dataDir,"estrogen-normalized.Rda"))
+ }
```

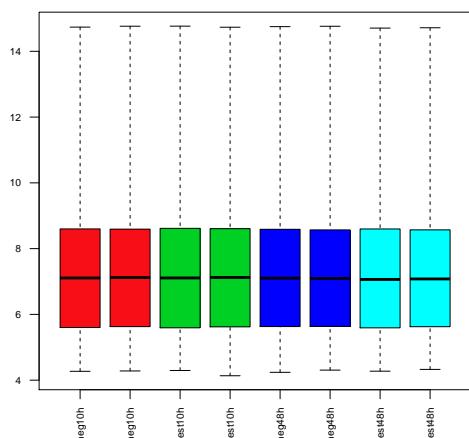
Background correcting
Normalizing
Calculating Expression

La normalización hace que los valores de los arrays sean comparables entre ellos, aunque los distintos métodos sitúan los valores en escalas distintas, por lo que lo que no resulta directamente comparable son los valores normalizados por distintos métodos.

Un boxplot de los valores normalizados sugiere que los valores ya están en una escala en donde se pueden comparar.

```
> boxplot(eset_rma,main="Boxplot for RMA-normalized expression values ",
+           names=sampleNames, cex.axis=0.7, col=info$grupo+1,las=2)
```

Boxplot for RMA-normalized expression values



El código siguiente, que no se ejecuta muestra como se podrían aplicar distintos métodos para luego compararlos entre ellos.

```
> eset_mas5 <- mas5(rawData) # Uses expresso (MAS 5.0 method) much slower than RMA!
> stopifnot(require(gcrma))
> eset_gcrma <- gcrma(rawData) # The 'library(gcrma)' needs to be loaded first.
> stopifnot(require(plier))
> eset_plier <- justPlier(rawData, normalize=T) # The 'library(plier)' needs to be loaded first.
> compara <- data.frame(RMA=exprs(eset_rma)[,1], MAS5 =exprs(eset_mas5)[,1],
+                         GCRMA=exprs(eset_gcrma)[,1], PLIER =exprs(eset_plier)[,1])
> pairs(compara)
```

Filtraje

El filtraje *no específico* permite eliminar los genes que varían poco entre condiciones o que deseamos quitar por otras razones como por ejemplo que no disponemos de anotación para ellos. La función `nsFilter` permite eliminar los genes que, o bien varían poco, o bien no se dispone de anotación para ellos.

Si al filtrar deseamos usar las anotaciones, o la falta de ellas, como criterio de filtraje debemos disponer del correspondiente paquete de anotaciones.

```
> require(genefilter)
> filtered <- nsFilter(eset_rma, require.entrez=TRUE,
+                         remove.dupEntrez=TRUE, var.func=IQR,
+                         var.cutoff=0.5, var.filter=TRUE,
+                         filterByQuantile=TRUE, feature.exclude="^AFFX")
```

La función `nsFilter` devuelve los valores filtrados en un objeto `expressionSet`

y un informe de los resultados del filtraje.

```
> names(filtered)

[1] "eset"      "filter.log"

> class(filtered$eset)

[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"

> print(filtered$filter.log)

$numDupsRemoved
[1] 2950

$numLowVar
[1] 4402

$numRemoved.ENTREZID
[1] 853

$feature.exclude
[1] 19

> eset_filtered <- filtered$eset
```

Podemos grabar el objeto `eset_rma` y los datos filtrados para su posterior uso.

```
> save(eset_rma, eset_filtered, file=file.path(resultsDir, "estrogen-normalized.Rda"))
```

Despues del filtraje han quedado 4409 genes disponibles para analizar.

9.4 Selección de genes diferencialmente expresados

Como en las etapas anteriores la selección de genes diferencialmente expresados (GDE) puede basarse en distintas aproximaciones, desde la *t* de Student al programa SAM pasando por multitud de variantes.

En este ejemplo se aplicará la aproximación presentada por Smyth *et al.* (2004) basado en la utilización del *modelo lineal general* combinada con un método para obtener una estimación mejorada de la varianza.

9.4.1 Análisis basado en modelos lineales

El capítulo 6 de tse manual o el manual del programa **limma** contienen explicaciones detalladas sobre construir un modelo lineal para este problema y como utilizarlo para seleccionar genes diferencialmente expresados.

Matriz de diseño

El primer paso para el análisis es crear la matriz de diseño.

La situación discutida en este ejemplo se puede modelizar de dos formas, tal como se discute en la presentación citada más arriba:

- Como un modelo de dos factores Estrogeno(Pres/Aus) Tiempo (10h/48h) con interacción.
- Como un modelo de un factor con cuatro niveles (Pres.10h/Pres.48h/Aus.10h/Aus.48h).

Tal como se describe en el manual de **limma** la segunda parametrización resulta a menudo más comoda, a pesar de parecer menos intuitiva, porque permite formular con más facilidad que la de dos factores las preguntas que típicamente interesan a los investigadores.

El modelo lineal para este estudio será:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Design Matrix, } \mathbf{X}} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{pmatrix}$$

Los parámetros del modelos representan las cuatro combinaciones tiempo/estrogeno.

$$\alpha_1 = \mathbf{E}(\log Abs.10h),$$

$$\alpha_2 = \mathbf{E}(\log Pres.10h),$$

$$\alpha_3 = \mathbf{E}(\log Abs.48h),$$

$$\alpha_4 = \mathbf{E}(\log Pres.48h).$$

La matriz de diseño puede definirse manualmente o a partir de un factor creado específicamente para ello.

Manualmente, seria:

```
> design.1<-matrix(
+ c(1,1,0,0,0,0,0,0,
+   0,0,1,1,0,0,0,0,
+   0,0,0,0,1,1,0,0,
+   0,0,0,0,0,0,1,1),
+ nrow=8,
+ byrow=F)
> colnames(design.1)<-c("neg10h", "est10h", "neg48h", "est48h")
> rownames(design.1) <- c("low10A", "low10B", "hi10A" , "hi10B", "low48A", "low48B", "hi48A" , "hi48B")
> print(design.1)
```

	neg10h	est10h	neg48h	est48h
low10A	1	0	0	0
low10B	1	0	0	0
hi10A	0	1	0	0
hi10B	0	1	0	0
low48A	0	0	1	0
low48B	0	0	1	0
hi48A	0	0	0	1
hi48B	0	0	0	1

Alternativamente puede crearse la matriz de diseño a partir de la información sobre las condiciones contenida en el `phenoData`, *siempre que exista un campo adecuado para ello*.

En este caso la columna `Target` se ha creado para utilizarla con esta finalidad.

El objeto `phenoData` puede recrearse a partir del archivo original o extrayéndolo del objeto `ExpressionSet` que contiene los datos y las covariables.

```
neg10h est10h neg48h est48h
low10A    1    0    0    0
low10B    1    0    0    0
hi10A     0    1    0    0
hi10B     0    1    0    0
low48A    0    0    1    0
low48B    0    0    1    0
hi48A     0    0    0    1
hi48B     0    0    0    1

attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$lev
[1] "contr.treatment"
```

Ambas matrices, `design`, `design.1` resultan iguales.

Contrastes

Dado un modelo lineal definido a través de una matriz de diseño pueden formularse las preguntas de interés como *contrastos* es decir comparaciones entre los parámetros del modelo.

Cada parametrización distinta requerirá de unos contrastes diferentes para las mismas preguntas, por lo que habitualmente se utilizará la parametrización que permita formular de forma más clara las comparaciones de interés.

En este caso interesa estudiar

- Efecto del estrógeno al inicio del tratamiento
- Efecto del estrógeno al cabo de un tiempo del tratamiento
- Efecto del tiempo en ausencia de estrógeno

Esto se puede formular facilmente con la parametrizacion adoptada.

$$\beta_1^1 = \alpha_2 - \alpha_1, \quad \text{Efecto del estrogeno pasadas 10 horas}$$

$$\beta_2^1 = \alpha_4 - \alpha_3, \quad \text{Efecto del estrogeno pasadas 48 horas}$$

$$\beta_3^1 = \alpha_3 - \alpha_1, \quad \text{Efecto del tiempo en ausencia de Estrogeno}$$

```
> cont.matrix <- makeContrasts (
+     Estro10=(est10h-neg10h),
+     Estro48=(est48h-neg48h),
+     Tiempo=(neg48h-neg10h),
+     levels=design)
> cont.matrix
```

Contrasts				
Levels	Estro10	Estro48	Tiempo	
neg10h	-1	0	-1	
est10h	1	0	0	
neg48h	0	-1	1	
est48h	0	1	0	

Estimación del modelo y selección de genes

Una vez definida la matriz de diseño y los contrastes podemos pasar a estimar el modelo, estimar los contrastes y realizar las pruebas de significación que nos indiquen, para cada gen y cada comparación, si puede considerarse diferencialmente expresado.

El método implementado en **limma** amplía el análisis tradicional utilizando modelos de Bayes empíricos para combinar la información de toda la matriz de datos y de cada gen individual y obtener estimaciones de error mejoradas.

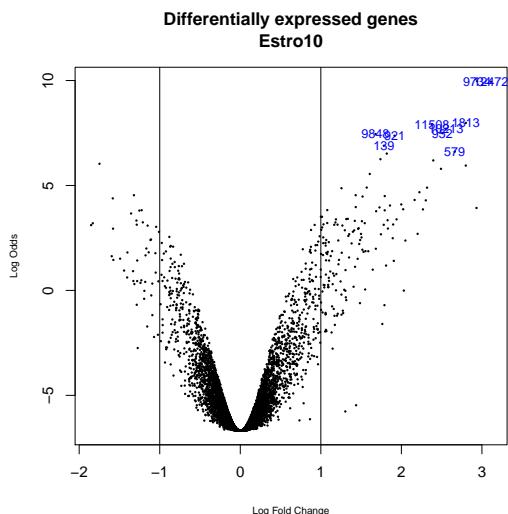
El análisis proporciona los estadísticos de test habituales como **Fold-change t**-moderados o *p*-valores ajustados que se utilizan para ordenar los genes de mas a menos diferencialmente expresados.

A fin de controlar el porcentaje de falsos positivos que puedan resultar del alto numero de contrastes realizados simultaneamente los *p*-valores se ajustan de forma que tengamos control sobre la tasa de falsos positivos utilizando el metodo de Benjamini y Hochberg ([4]).

La funcion **topTable** genera para cada contraste una lista de genes ordenados de mas a menos diferencialmente expresados.

```
> topTabEstro10 <- topTable (fit.main, number=nrow(fit.main), coef="Estro10", adjust="fdr")
> topTabEstro48 <- topTable (fit.main, number=nrow(fit.main), coef="Estro48", adjust="fdr")
> topTabTiempo <- topTable (fit.main, number=nrow(fit.main) , coef="Tiempo", adjust="fdr")
```

Una forma de visualizar los resultados es mediante un **volcano plot** que representa en abscisas los cambios de expresión en escala logarítmica y en ordenadas el “menos logaritmo” del p-valor o alternativamente el estadístico *B*.



9.4.2 Comparaciones múltiples

Cuando se realizan varias comparaciones a la vez puede resultar importante ver que genes cambian simultáneamente en más de una comparación. Si el número de comparaciones es alto también puede ser necesario realizar un ajuste de p-valores entre las comparaciones, distinto del realizado entre genes.

La función **decidetests** permite realizar ambas cosas. En este ejemplo no se ajustaran los p-valores entre comparaciones. Tan solo se seleccionaran los genes que cambian en una o más condiciones.

EL resultado del análisis es una tabla **res** que para cada gen y cada comparación contiene un 1 (si el gen esta sobre-expresado o “up” en esta condicion), un 0 (si no hay cambio significativo) o un -1 (si esta “down”-regulado).

Para resumir dicho análisis podemos contar qué filas tienen como mínimo una celda distinta de cero:

```
> sum.res.rows<-apply(abs(res),1,sum)
> res.selected<-res[sum.res.rows!=0,]
> print(summary(res))
```

```
> vennDiagram (res.selected[,1:3], main="Genes in common #1", cex=0.9)
```

Genes in common #1

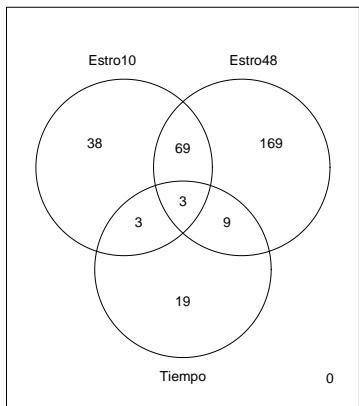


Figura 49: Número de genes seleccionado en cada comparación

	Estro10	Estro48	Tiempo
-1	23	99	27
0	12512	12375	12591
1	90	151	7

Un diagrama de Venn permite visualizar la tabla anterior sin diferenciar entre genes “up” o “down” regulados.

9.4.3 Anotación de resultados

La identificación de los genes seleccionados puede resultar más sencilla para el especialista en un campo si se utilizan nombres estándar como el simbolo del gen o “gene symbol”. Con esta finalidad cada paquete de anotaciones tiene tablas de correspondencia entre los distintos tipos de identificadores, principalmente entre los del array y los de otras bases de datos.

Para saber que anotaciones estan disponibles debe cargarse el paquete y llamar la función del mismo nombre

```
> require(hgu95av2.db)
> hgu95av2()
```

Bioconductor dispone de algunos paquetes que permiten aprovechar esta funcionalidad anterior para obtener las anotaciones de cada gen y generar una tabla HTML con enlaces a algunas bases de datos.

De forma sencilla es posible obtener tablas con las anotaciones correspondientes a los genes seleccionados. Si se desea ser más ambicioso es posible generar tablas en las que se combinen hiperenlaces a las anotaciones con los resultados de la selección de genes.

Tablas de anotaciones sencillas

El paquete **annaffy** permite de forma muy simple generar una tabla de anotaciones con hiperenlaces a las bases de datos para cada anotación seleccionada.

La instrucción siguiente crea una tabla con las anotaciones disponibles para los genes seleccionados en la sección de comparaciones múltiples.

```
> require(annaffy)
> genesSelected <- rownames(res.selected)
> at <- aafTableAnn(genesSelected, "hg19av2.db")
> saveHTML(at, file.path(resultsDir, "anotations.html"),
+           "Annotations for selected genes")
```

9.4.4 Visualización de los perfiles de expresión

Tras seleccionar los genes diferencialmente expresados podemos visualizar las expresiones de cada gen agrupándolas para destacar los genes que se encuentran up o down regulados simultáneamente constituyendo *profiles de expresión*.

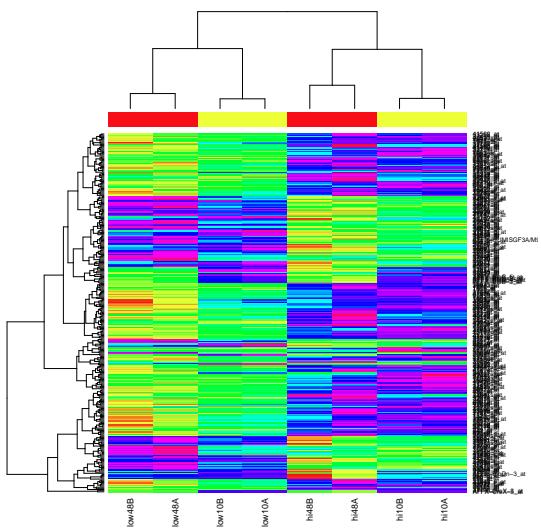
Hay distintas formas de visualización pero aquí tan sólo se presenta el uso de mapas de color o **Heatmaps**.

En primer lugar seleccionamos los genes a visualizar: Se toman todos aquellos que han resultado diferencialmente expresados en alguna de las tres comparaciones.

```
> probeNames<-rownames(res)
> probeNames.selected<-probeNames[sum(res.rows)!=0]
> exprs2cluster <-exprs(eset_rma)[probeNames.selected,]
```

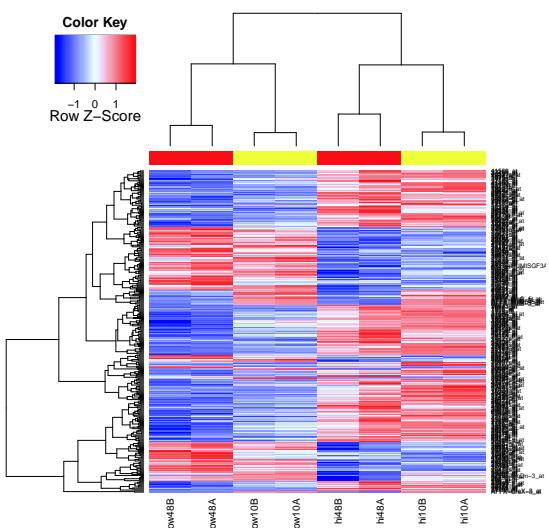
Para representar el Heatmap tan sólo necesitamos la matriz de datos resultante.

```
> color.map <- function(horas) { if (horas< 20) "yellow" else "red" }
> grupColors <- unlist(lapply(pData(eset_rma)$time.h, color.map))
> heatmap(exprs2cluster, col=rainbow(100), ColSideColors=grupColors, cexCol=0.9)
```



Si se desea realizar mapas de color más sofisticados puede utilizarse el paquete `gplots` que implementa una versión mejorada en la función `heatmap.2`

```
> color.map <- function(horas) { if (horas< 20) "yellow" else "red" }
> grupColors <- unlist(lapply(pData(eset_rma)$time.h, color.map))
> require("gplots")
> heatmap.2(exprs2cluster,
+             col=bluered(75), scale="row",
+             ColSideColors=grupColors, key=TRUE, symkey=FALSE,
+             density.info="none", trace="none", cexCol=1)
```



10. Métodos avanzados: Busca de patrones y predicción

10.1 Descubrimiento de grupos en datos de microarrays

El análisis de grupos (“clusters”), conocido también como “class discovery”, fue, en los primeros tiempos de los microarrays el método más popular de entre los empleados para tratar de identificar y agrupar genes expresados de forma similar y correlacionar los resultados con los procesos biológicos.

La idea subyacente en esta aproximación era (es) que cuando dos genes estén corregulados y por lo tanto, funcionalmente relacionados, es probable que se expresen de forma simultánea, es decir, cuando la expresión de uno de ellos aumente, también aumentará la del otro, con lo que será posible agruparlos.

La figura 50 es un ejemplo de como 6 genes distintos presentan perfiles de expresión similares a lo largo de múltiples condiciones experimentales en un experimento de respuesta al estrés en levadura.

Figura 50. Perfiles de expresión

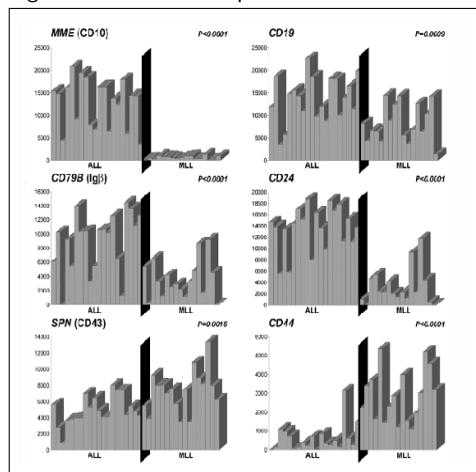


Figura 50

Perfiles de expresión similares de 6 genes entre dos formas de leucemia (ALL y AML)

A partir de la aplicación de un método de agrupación suele ser posible la representación de los grupos en un plano con lo que se consigue una representación sencilla de una matriz de datos de alta dimensión que permite visualizar los datos de forma intuitiva, de forma parecida a lo que permite el Análisis de Componentes Principales.

El análisis de conglomerados se ha aplicado tanto a la clasificación de genes (o variables) como de arrays (o muestras). Básicamente la clasificación de los genes persigue:

- la identificación de grupos de genes corregulados,
- la identificación de patrones de expresión espacial o temporal o
- la reducción de la redundancia en modelos predictivos.

Si por lo contrario, se agrupan las muestras , el resultado nos permitirá:

- Identificar nuevas clases biológicas (p.e. nuevas clases de tumor).
- Detectar artefactos experimentales.

Usualmente se realiza la agrupación simultánea de los arrays y los genes, es decir de las columnas y las filas de la matriz de expresión tal como se muestra en la figura 51.

Figura 51. Agrupación simultánea de genes y muestras

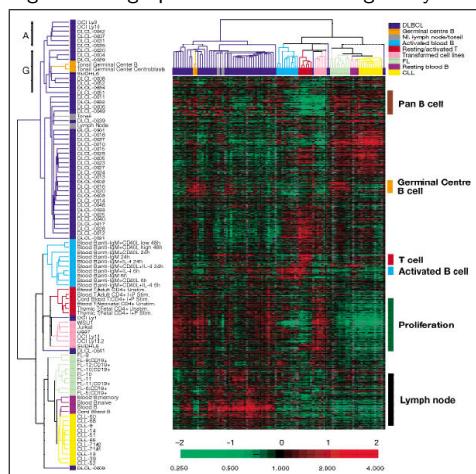


Figura 51

Ejemplo de un clustering simultáneo de arrays y genes .Ver [1]

10.1.1 Principios y métodos para el descubrimiento de clases

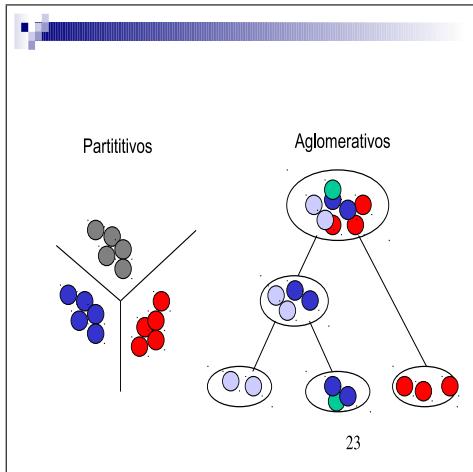
El análisis de conglomerados es una técnica estadística clásica, o mejor dicho un amplio abanico de métodos y técnicas cuyo rasgo común es la búsqueda de patrones en los datos que permitan juntar los más similares entre si diferenciándolos de los más heterogéneos.

Prácticamente todos los métodos de agrupación se han utilizado de una forma u otra para el análisis de datos de microarrays pero sin duda los métodos más populares son el el cluster jerárquico entre los métodos aglomerativos y el método de las *k*-medias entre los divisivos. La diferencia entre ambas aproximaciones se muestra en la figura 52

Lectura Complementaria

Se puede obtener más información de cómo se realiza una agrupación simultánea de arrays y genes en [1]

Figura 52. Métodos aglomerativos y divisivos.



El análisis de conglomerados siempre considera dos aspectos complementarios:

- La medida de *distancia* que se utilizará para cuantificar la similaridad entre filas o columnas.
- El algoritmo de agrupación con el que se determinaran que datos son similares o cuales no lo son.

A continuación se repasan brevemente cada una de estas componente centrándose en sus aplicaciones a los microarrays.

Figura 52

Los métodos aglomerativos parten de una situación en que cada grupo inicial está formado por un solo elemento y en el proceso los va agrupando hasta llegar a un único grupo formado por todos los elementos. En los métodos divisivos se parte de un grupo con todos los elementos que se va dividiendo en subgrupos.

1 Medidas de distancia

Dados dos valores \mathbf{x} , \mathbf{y} de un espacio n dimensional una distancia es una función de dichos puntos que permite cuantificar su similaridad.

Existen multitud de distancias, algunas de las más conocidas son:

- La distancia euclídea

$$d_{Euc}(\mathbf{x} - \mathbf{y}) = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$$

- La distancia valor absoluto o distancia “ciudad”

$$d_{abs}(\mathbf{x} - \mathbf{y}) = \sum_{i=1}^n |x_i - y_i|$$

- La distancia “correlación”

$$d_{cor}(\mathbf{x} - \mathbf{y}) = 1 - cor(\mathbf{x}, \mathbf{y})$$

donde $cor(\mathbf{x}, \mathbf{y})$ representa el coeficiente correlación lineal entre \mathbf{x} e \mathbf{y} .

La elección de la distancia es un tema importante que influye en los resultados de la agrupación. Por ejemplo la distancia euclídea detecta bien diferencias absolutas mientras que la correlación es mejor para detectar cambios de tendencia (si unos “suben” u otros “bajan” esta distancia será pequeña independientemente de la diferencia entre unos y otros).

2 Algoritmos de agrupación

- Métodos jerárquicos: El cluster jerárquico descrito en el capítulo 3 es sin duda la herramienta más conocida entre los usuarios de los microarrays. Fue popularizado por los primeros estudios realizados por Eisen [18] quien, además proporcionaba un programa libre para agrupar y visualizar genes lo que le dio una gran difusión. Este método se ha utilizado principalmente para la agrupación de las muestras más que de los genes, ya que al ser su número mucho menor facilita la interpretación del dendrograma obtenido.

Lectura Complementaria

Eisen(1998)

Aunque los métodos jerárquicos son superados en calidad y precisión por muchas otras técnicas de agrupación existe una técnica de visualización que hace que su uso continue. Se trata de los mapas de color o “heatmaps” (ver Gentleman y otros [22], capítulo 10) que consiste en un rectángulo dividido en filas y columnas que constituyen bloques que están coloreados, donde el color de cada uno de estos bloques representa en nivel de expresión de un gen en un array (ver figura ??).

Típicamente en un heatmap se emplean distintas tonalidades de rojo para representar diferentes grados de incremento en la expresión, mientras que las tonalidades de color verde se usan para representar los distintos grados de decrecimiento de la expresión. Aunque esta elección de colores es bastante estándar, se puede modificar utilizando otras combinaciones de colores.

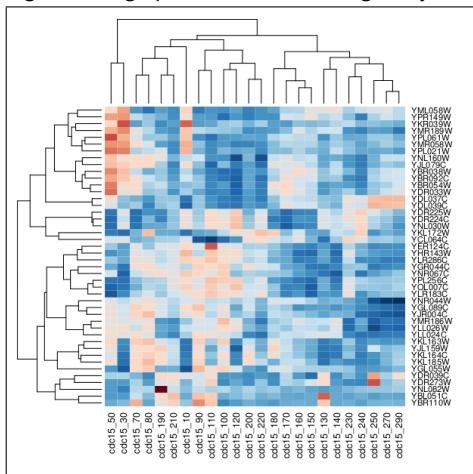
En un heatmap cada columna representa un array y cada fila se corresponde con un gen. Si se aplica un cluster jerárquico a las filas y a las columnas se obtiene una visualización que permite detectar con relativa facilidad posibles grupos de genes y muestras con patrones de expresión similares, es decir cuya expresión aumenta o disminuye simultáneamente de forma similar.

- El método *k-means*: Este método (ver Kaufman & Rosseeuw [29] es así mismo

Figura 53

Un heatmap permite detectar mediante un cluster jerárquico genes y muestras con patrones de expresión similares

Figura 53. Agrupación simultánea de genes y muestras en un heatmap



muy popular a pesar de tener la desventaja de que es un algoritmo que comienza con una muestra de “ k ” genes elegidos al azar de la matriz original de datos. Cada uno de ellos se utiliza como el centroide inicial de los “ k ”clusters que se van a formar, lo que hace que el resultado final sea muy sensible a estas elecciones. Cada cluster inicial esta representado por un elemento llamado *centroide*. En este caso el investigador debe probar con distintas cantidades de clusters iniciales(k) y a continuación seleccionar la k que mejor se ajusta a los datos.

Los dos métodos de clustering comentados presentan un problema que resulta difícil de resolver, y es que el orden de los genes para un cluster dado y el orden en que se muestran los clusters no aporta ningún tipo de información biológica útil.

Esto implica que algunos clusters que quedan representados cerca unos de otros pueden ser menos parecidos entre sí que con otros clusters que aparecen en posiciones más lejanas.

- Otros métodos de agrupación: Entre la multitud de métodos existentes podemos destacar *Partition Around Medoids* (PAM) similar al $K - means$ pero más robusto y sin inicialización aleatoria o los mapas auto–organizativos (*Self-Organizing Maps*(SOM) [12] que se han aplicado con bastante éxito a datos de microarrays.

Los mapas auto-organizados (“Self-Organising Maps”) se obtienen a partir de la aplicación de los métodos basados en redes neuronales. El algoritmo permite, de forma iterativa, que los patrones mas parecidos se vayan juntando entre si y alejándose de aquellos otros que son mas diferentes. Este tipo de algoritmos son mas fiables y robustos puesto que se basan en redes neuronales que por definición son capaces de trabajar con grandes cantidades de datos con ruido. Sin embargo, no carece de ciertos inconvenientes. SOM es una herramienta particularmente útil en el tratamiento de datos procedentes de series temporales. De cualquier modo cada uno de ellos tiene sus propios inconvenientes, y

para muchos usuarios la agrupación jerárquica sigue siendo la opción favorita.

10.1.2 Consistencia de la agrupación

Número de clusters

Un problema complejo en el análisis de clusters es el saber *cuantos clusters existen realmente*, dado que, debemos recordar que la pertenencia de un individuo a un cluster no está fijada de antemano sino que la establece el algoritmo. Se trata de un problema importante pero complejo y muchos autores han propuesto diversas aproximaciones para resolverlo (veáse, por ejemplo Milligan and Cooper [38]).

Uno de los enfoques más populares es el del gráfico de silueta (“Silhouette plot” introducido por Rousseeuw ([42]) o el de la silueta promedio “Average Silhouette” donde Kaufman y Rousseeuw [29] amplian lo previamente publicado.

Algunos de estos métodos se han aplicado con éxito a datos de microarray. En otros casos se han desarrollado extensiones específicas para adaptarse mejor a sus particularidades:

- Yeung et al. [54] y Mc Lachlan propusieron distintos tipos de métodos basados en modelos.
- Hastie y colegas [25] introdujeron el estadístico “GAP” como base para la selección del número de clusters.
- Dudoit & Fridlyand, [13] propusieron un algoritmo llamado *Clest* que usa el remuestreo para estimar la cantidad de clusters en base a la precisión de la predicción. El método se puede usar con *cualquier* algoritmo de agrupamiento y parece estar mejor adaptado para muestras agrupadas que para genes agrupados.

Validación de las agrupaciones

Cuando se realiza un clustering de muestras el dendrograma puede dar una idea acerca de la similaridad y relación entre las muestras, pero no nos indica la robustez del método frente a la variabilidad asociada al proceso de muestreo.

Con el fin de extraer conclusiones válidas de la estructura de la agrupación subyacente en los datos es necesario investigar cómo afecta la variabilidad a los resultados del análisis del cluster.

Por otro lado, la evaluación de la validez de los clusters obtenidos es especialmente importante cuando se agrupan datos de microarrays. El hecho de que las

proteínas se organízen en vías y los genes estén co-regulados sugiere que el perfil de expresión de un gran número de genes poseen una estructura de agrupación natural. Por lo tanto existen una serie de clusters “reales” que deberían ser descubiertos.

La dificultad en la validación de clusters es que no hay una clasificación inicial contra la que se puedan comparar los resultados del clustering. Una forma de abordar este problema es examinar las relaciones entre los resultados del clustering y algunas variables externas que no hayan sido utilizadas previamente, siempre y cuando ello sea posible. Además se pueden haber generado algunos clusters que no tengan relación con estas variables.

Un enfoque común para evaluar la validez del cluster es utilizar algún tipo técnica de remuestreo, tal como el método de bootstrap desarrollado por Kerr y Churchill ([31]).

La “Figure of Merit” (FOM) introducida por Yeung y colegas ([54]) es otro enfoque puramente experimental ampliamente utilizado en otros contextos, que ha sido validado específicamente para datos de microarrays.

Comentarios finales

Hasta el momento se ha considerado la utilización de los métodos de la agrupación para encontrar grupos de genes co-regulados o muestras relacionadas que pertenecen a grupos no identificados previamente.

En concreto podemos encontrar una agrupación de muestras que sugiere que –donde inicialmente no parecían haber subgrupos– que han aparecido grupos asociados de alguna forma al proceso en estudio(es decir obtenemos agrupaciones de muestras/individuos que no conocíamos previamente). Este tipo de agrupación se realiza principalmente con el conjunto de genes en el cual se ha observado un cambio, por ejemplo, los genes diferencialmente expresados .

Sin embargo, existe otra aplicación importante del clustering de muestras, que se realiza en todos los genes, no sólo en aquellos que han sido seleccionados. El objetivo sería agrupar los datos iniciales (normalizados) para descubrir patrones, probablemente debido a algún efecto sistemático (bloque). Puede haber múltiples fuentes de esta variación sistemática: lotes de producción, técnicas, de origen biológico (líneas celulares), etc. Una visualización adecuada de los glusters obtenidos puede ayudar a descubrir la existencia de estos efectos.

Después de detectar tales efectos no esperados, es posible incluirlos en el modelo utilizado para la detección de genes expresados diferencialmente por lo que se pueden estimar su efecto de forma separada de los factores importantes de manera que no afecten a los resultados finales del análisis.

10.2 Diagnósticos moleculares y métodos de clasificación

El objetivo de la predicción de una clase o grupo (en análisis de microarrays como en la mayoría de los problemas de clasificación) es desarrollar una función multivariante para predecir con exactitud la pertenencia de clase (fenotipo) de un nuevo individuo.

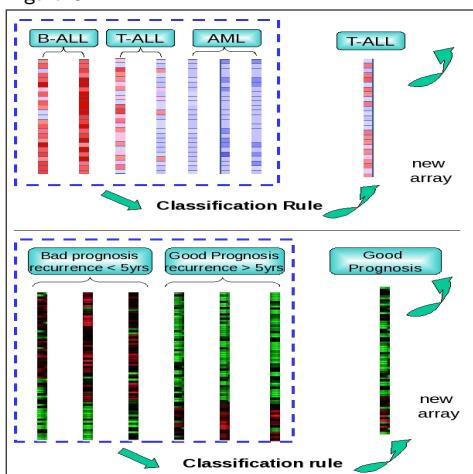
Es decir, si cada objeto (paciente, muestra, chip) i se le asigna:

- una etiqueta de clase (o respuesta) $Y \in \{1, 2, \dots, K\}$ y
- un vector de características de las variables de predicción (medición de la expresión de g genes), $\mathbf{X} = (X_1, \dots, X_g)$,

el objetivo consiste en obtener la predicción de $Y(\mathbf{X})$ para un nuevo individuo sin clasificar. Desde el punto de vista biomédico, es importante distinguir entre los *predicción de clase* (asignación de una nueva muestra a categorías existentes) y *pronóstico* (predicción de la evolución de la enfermedad de un paciente).

Un ejemplo de predicción es el estudio de clasificación molecular de pacientes afectados de distintas variantes de leucemia realizado por Golub y colegas [24] (ver figura 54, mientas que un ejemplo de pronóstico puede ser la construcción de un indicador para determinar qué tumores pueden desarrollar la metástasis después de un cierto período de tiempo estudiado por Van't Veer y colegas [50] (ver figura 54, b)).

Figura 54.



Esta sección está organizada de la siguiente manera: en primer lugar se presentan brevemente algunos de los principales métodos de clasificación, con énfasis en su aplicación al análisis de microarrays. A continuación se discute el problema de la selección de variables para la predicción y se dan algunas ideas sobre cómo medir el rendimiento de un clasificador. Finalmente se concluye con una breve discusión sobre algunas cuestiones específicas de la predicción de clase con datos de expresión y reproduciendo algunas "consejos prácticos" para los usuarios.

Figura 54

Dos ejemplos para ilustrar la clasificación de problemas en el análisis de los datos de microarrays. (a) Ejemplo de predicción de clase: Asignación del tipo de patología a un nuevo paciente. Tomado de [24]. (b) Predicción de desarrollo de metástasis

Esta sección se basa principalmente en el material de Diaz Uriarte [11].

Métodos de predicción de clase El número de métodos de clasificación disponible es muy alto, probablemente debido al hecho de que es un término muy general que puede abarcar desde una “simple regresión” logística a un complejo sistema de máquinas de vectores soporte multicategóricas.

- **Análisis discriminante:** Uno de los métodos más populares entre los estadísticos es sin duda el *análisis discriminante* [10], que permite clasificar los resultados binarios o múltiples usando una función lineal o cuadrática – llamada función discriminante– de las variables continuas que, bajo supuestos de normalidad, se puede obtener mediante maximización de la razón de la verosimilitud entre grupos frente a la verosimilitud dentro de los grupos.

En análisis de microarrays se han utilizado con éxito dos variantes del análisis discriminante. Uno de ellos es *Análisis Discriminante Lineal Diagonal(DLDA)* que proporciona la discriminación óptima cuando matriz de varianzas–covarianzas se supone diagonal. Otra és el algoritmo de la ponderación de los votos “weight voting algorithm” introducidos por Golub y colegas ([24]), que se ha convertido en relativamente popular y viene a ser una variante de DLDA ([15])

- **Métodos de vecino más próximo:** Los métodos del *vecino más cercano* o de “Nearest neighbor” (KNN) también se han utilizado profusamente probablemente debido a su simplicidad. Estos métodos consisten en predecir el grupo de un caso de prueba mediante la mayoría de votos entre los (k) vecinos más cercanos a dicho caso. El método es sencillo, intuitivo y no realiza ninguna suposición sobre los datos.
- **Las máquinas de vectores soporte :** Estos métodos, tambien denominados “Support Vector Machines” (SVM), provenientes del campo del aprendizaje automático o “machine learning” han tenido también bastante éxito en el campo de los microarrays a pesar de que, en contraste con los métodos del vecin más próximo no son en absoluto intuitivos. La idea de los SVM es que si no puede separar bien dos grupos en un espacio lo intenta proyectando los valores a un espacio de mayor dimensión hasta obtener el mejor hiperplano de separación entre las clases definido como aquel que garantiza hay una distancia máxima entre el hiperplano y el punto más cercano de cualquiera de las clases. Incluso cuando no hay hiperplano de separación SVMs puede producir clasificadores decentes tratando de maximizar el margen y permite algunos errores de clasificación sujeta a la restriccción de que el error total (distancia del hiperplano en el “lado equivocado”) es menor que una constante. La flexibilidad y versatilidad de la SVM ha hecho una opción muy popular entre los profesionales, pero su característica de “caja negra”, así como el hecho de que son más difíciles de entender que los enfoques más simples probablemente ha restringido su extensión.

- **Otros métodos** Existen muchos métodos más, desde métodos estadísticos tradicionales, como la *regresión logística* a métodos modernos más sofisticados, como los “random forests”, para no hablar de todos los métodos desarrollados *ad hoc* para el análisis de datos de expresión como es el Análisis de Predicción de Microarrays (PAM) o el de “gene shaving”.

En el problema de la predicción de la clase en microarrays, como es el caso en otros campos, se ha hecho un uso extensivo de los métodos de agregación, esto es la combinación de varios predictores para obtener clasificadores mejorados. La agregación se sugirió por primera vez por Breiman ([5]), quien encontró que el aumento en la precisión se podría obtener por agregación de factores predictivos construidos a partir versiones perturbadas del conjunto de aprendizaje. Bagging ([5]) y Boosting ([21]) son dos métodos de agregación basados en el remuestreo que se han aplicado con éxito relativo en microarrays.

Comparación entre los métodos Dado el número y la diversidad de los métodos disponibles una de las primeras preocupaciones de un usuario potencial de los métodos de predicción de la clase es cuál debe utilizar para cada problema.

Para ayudar a responder esta pregunta Dudoit [14] hizo una comparación de varios métodos de clasificación populares. Su principal conclusión fue que *los clasificadores simples, tales como análisis discriminante lineal Diagonal (DLDA) y el vecino más cercano (NN) funcionan notablemente bien en comparación con los más sofisticados, como los árboles de la clasificación agregada*.

Won Lee y colegas [52] ampliaron el análisis anterior incluyendo más métodos (hasta 21) y más conjuntos de datos (7). En algunos aspectos alcanzaron conclusiones similares a las de [14], aunque acabaron argumentando que los métodos más complejos como las máquinas soporte vector obtenían un mejor rendimiento.

A pesar de todo lo dicho, después de más de 10 años de aplicaciones en este campo, todavía no existe un consenso claro sobre cual es el método más adecuado para cada caso. Si que hay, sin embargo, un cierto consenso sobre que se debe evitar. Allison ([2] o Dupuy y Simon [16] enumeran algunas buenas prácticas que conviene seguir cuando se utilizan los métodos de predicción. Recientemente se ha llevado a cabo un gran estudio el “MAQC-II” (“Micro Arrays Quality Control-II”, [9]) del que se han derivado también algunas conclusiones sobre que es lo que influye, positiva o negativamente, en la calidad y fiabilidad de un predictor hecho con datos de microarrays.

10.2.1 La selección de variables para la clasificación

Con el fin de construir un predictor uno debe decidir qué variables a utilizar. Esto no es un problema trivial, ya que esta selección va a guiar todo el proceso e

influirá considerablemente en los resultados. Si el número de variables es pequeño no es difícil elegir entre ellas o simplemente utilizarlos todos. Sin embargo, cuando hay miles de variables para escoger la selección se convierte en una difícil labor.

Algunos métodos, como SVM, KNN o DLDA pueden utilizar tantas variables como se les suministre. Otros métodos, tales como la regresión logística, o regresión de Cox no pueden, debido a lo que se conoce como la “maldición de la dimensionalidad” o problema “ $p >> n$ ” que es el hecho de que el número de variables (=funciones=genes) (p) es mucho mayor que el número de muestras (n). En cualquier caso, el uso de algún procedimiento para la pre-selección de genes se considera que beneficia el rendimiento de la predicción.

Existen muchos métodos de selección pero no se discutirán aquí. Una buena revisión se encuentra disponible en [34].

Evaluación del clasificador Siempre es posible, dados unos datos y unas etiquetas de clase, construir un clasificador que asigne cada individuo a una clase. Sin embargo, en la práctica lo que se necesita es alguna garantía de que el clasificador que se va obtener es lo suficientemente bueno, asumiendo que la obtención del predictor “óptimo” suele ser imposible salvo en casos puntuales.

Para decidir la calidad de un predictor solemos basarnos en dos conceptos:

- Su *discriminabilidad*, es decir, lo bien que predice observaciones “nuevas” obtenidas de forma independiente de aquellas con las que se ha construido el predictor,
- Su *fiabilidad* entendido como la capacidad de proporcionar previsiones (clasificaciones) similares cuando los datos presentan (pequeñas) variaciones no atribuibles a los grupos que se clasifican.

En la práctica, muchos usuarios se basan en algún tipo de tasa de error para evaluar la discriminabilidad el predictor, es decir en el porcentaje de clasificación errónea frente al de predicciones correctas obtenidas por el clasificador.

Otro aspecto importante es que cualquier regla de clasificación tiene que evaluarse no por su capacidad de predecir las observaciones con que ha sido desarrollada sino por cómo predice nuevas observaciones independientes de las anteriores.

Ahora bien, en la práctica casi nunca se dispone de un grupo de datos independiente con el que evaluar el clasificador por lo que se re-utilizan los datos originales mediante algun procedimiento –que se denomina de validación cruzada–que permita simular de alguna forma la independencia entre las muestras. El método más conocido es el “leave-one-out (LOO), consistente en, si se tiene una muestra de n individuos, construir n predictores con $n - 1$ individuos, dejando uno “fuera”

cada vez y clasificándolo con la regla construida sobre los $n - 1$ restantes.

La recomendación de la mayoría de los expertos como Simon y colegas ([46]) es que, no sólo es preciso utilizar algún sistema de validación cruzada, sino que éste debe integrarse en *todo el proceso de construcción del predictor*, es decir no debe realizarse validación cruzada en una etapa y en otra no sino que todas las fases de construcción y validación deben someterse a este proceso.

Las figuras 55 y 56, inspiradas en Dupuy ([16] muestran de forma simplificada dos esquemas habituales de validación cruzada.

Figura 55. Esquema de validación “leave-one-out”

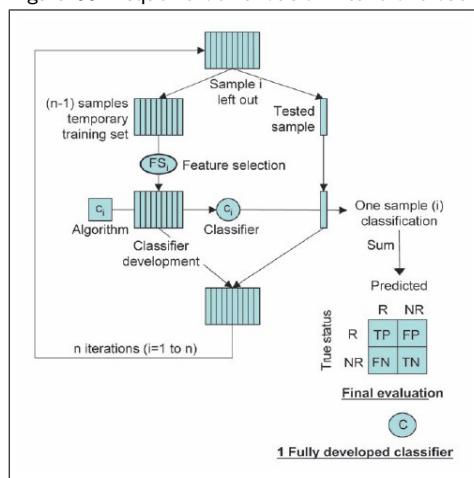


Figura 55

La validación “leave-one-out” elimina un individuo de la muestra, construye el predictor con los restantes y clasifica el individuo eliminado. Este proceso se repite n veces y el porcentaje de malas clasificaciones estima la fiabilidad del clasificador.

Figura 56. Esquema de validación “Random Split”

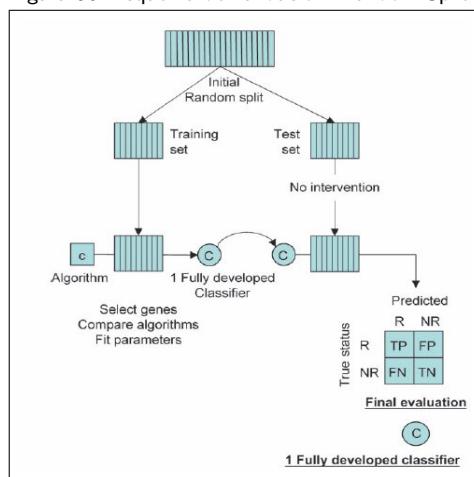


Figura 56

La validación “Random-split” divide los datos originales en dos grupos -escogidos aleatoriamente- uno se utiliza para construir el predictor y el otro para estimar la probabilidad de error. Este proceso se repite muchas veces y el promedio de malas clasificaciones estima la fiabilidad del clasificador.

11. Caso resuelto: Descubrimiento de clases

11.1 Clustering y visualización

En este caso se presentan distintos métodos de agrupación (“clustering”) disponibles en R y Bioconductor. Se utilizará el conjunto de datos **yeast** que puede descargarse desde la página del proyecto “Yeast Cell Cycle Project” (Proyecto de estudio del ciclo celular de la levadura) en la dirección:

<http://genome-www.stanford.edu/cellcycle/data/rawdata/>

Para descargarlo se tiene que acceder a la dirección mencionada y, desde ella, seleccionar el enlace “Tab delimited data” y descargar el archivo **combined.txt**.

11.1.1 Pre-procesamiento de los datos

La agrupación de datos se suele aplicar después del pre-procesamiento y filtrado de los mismos. El conjunto de datos de ejemplo ya ha sido normalizado y filtrado por lo que no nos ocuparemos de este aspecto.

El primer paso, como de costumbre, será establecer el directorio de trabajo que contiene los datos. El archivo de datos contiene información de varios experimentos (**factor alfa**, **cdc15** y **cinéticas de decantación**). Para simplificar en este ejercicio nos ocuparemos únicamente de los datos denominados “**cdc15**” (“cell division control protein 15”).

EL primer paso consiste en extraer los valores del grupo **cdc15** del conjunto de datos pre-normalizados y eliminar los genes con valores faltantes:

```
> workingDir <- getwd()
> dataDir <- file.path(workingDir, "data")
> d <- read.table(file.path(dataDir, "combined.txt"), sep="\t", header=T)
> names(d)

[1] "X"          "cln3.1"      "cln3.2"      "clb"         "clb2.2"      "clb2.1"
[7] "alpha"       "alpha0"       "alpha7"       "alpha14"      "alpha21"      "alpha28"
[13] "alpha35"     "alpha42"     "alpha49"     "alpha56"      "alpha63"      "alpha70"
[19] "alpha77"     "alpha84"     "alpha91"     "alpha98"      "alpha105"     "alpha112"
[25] "alpha119"    "cdc15"      "cdc15_10"   "cdc15_30"   "cdc15_50"   "cdc15_70"
[31] "cdc15_80"   "cdc15_90"   "cdc15_100"  "cdc15_110"  "cdc15_120"  "cdc15_130"
```

```
[37] "cdc15_140" "cdc15_150" "cdc15_160" "cdc15_170" "cdc15_180" "cdc15_190"
[43] "cdc15_200" "cdc15_210" "cdc15_220" "cdc15_230" "cdc15_240" "cdc15_250"
[49] "cdc15_270" "cdc15_290" "cdc28"      "cdc28_0"    "cdc28_10"   "cdc28_20"
[55] "cdc28_30"   "cdc28_40"   "cdc28_50"   "cdc28_60"   "cdc28_70"   "cdc28_80"
[61] "cdc28_90"   "cdc28_100"  "cdc28_110"  "cdc28_120"  "cdc28_130"  "cdc28_140"
[67] "cdc28_150"  "cdc28_160"  "elu"       "elu0"     "elu30"    "elu60"
[73] "elu90"     "elu120"   "elu150"   "elu180"   "elu210"   "elu240"
[79] "elu270"   "elu300"   "elu330"   "elu360"   "elu390"
```

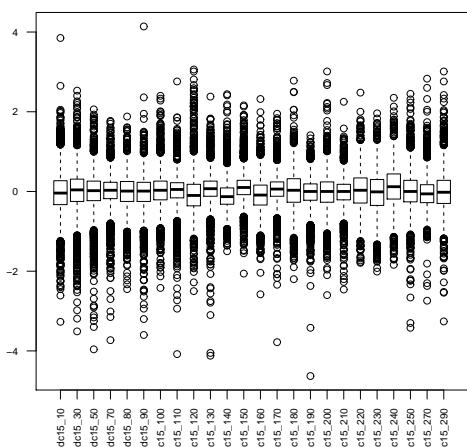
```
> cdc15 <- which(substr(names(d),1,6)=="cdc15_")
> dat <- d[,cdc15]
> names(dat)
```

```
[1] "cdc15_10"  "cdc15_30"  "cdc15_50"  "cdc15_70"  "cdc15_80"  "cdc15_90"
[7] "cdc15_100" "cdc15_110" "cdc15_120" "cdc15_130" "cdc15_140" "cdc15_150"
[13] "cdc15_160" "cdc15_170" "cdc15_180" "cdc15_190" "cdc15_200" "cdc15_210"
[19] "cdc15_220" "cdc15_230" "cdc15_240" "cdc15_250" "cdc15_270" "cdc15_290"
```

```
> rownames(dat) <- d$X
> dat <- na.omit(dat)
```

Los datos descargados de la web habían sido previamente normalizados. Para comprobarlo podemos realizar un boxplot y comprobar que los datos presentan una distribución similar y centrada en el cero como sería de esperar con valores de expresión relativa normalizados.

```
> boxplot(dat, las=2, cex.axis=0.7)
```



A continuación seleccionaremos sólo aquellos genes que se encuentran entre el 1% de las desviaciones estándar más altas:

```
> percentage <- c(0.99)
> sds <- apply(dat, MARGIN=1, FUN="sd")
> sel <- (sds>quantile(sds,percentage))
> dat.sel <- dat[sel, ]
> dim(dat.sel)
```

```
[1] 44 24
```

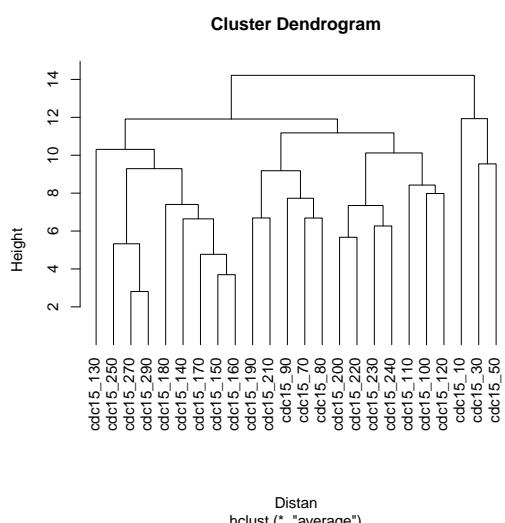
11.1.2 Agrupación jerárquica de las muestras

La primera aproximación que suele hacerse para establecer la relación entre entre muestras es la aplicación de algun método de agrupamiento jerárquico. En este caso se realizará un cluster jerárquico basado en distancias euclídeas y enlaces promedio ("average linkage").

```
> distmeth <- c("euclidean")
> Distan <- dist(t(dat.sel), method=distmeth)
> treemeth <- c("average")
> hc <- hclust(Distan, method=treemeth)
```

Para representar la estructura jerárquica de un agrupamiento se utiliza el dendrograma, un gráfico que tiene forma de árbol invertido, donde los nombres de los genes equivaldrían a las hojas.

```
> plot(hc, hang=-1)
```



Esta no es la única forma de realizar este tipo de agrupación y, en general se recomienda probar con distintos métodos “razonables” y ver hasta qué punto cambian los resultados que se obtiene.

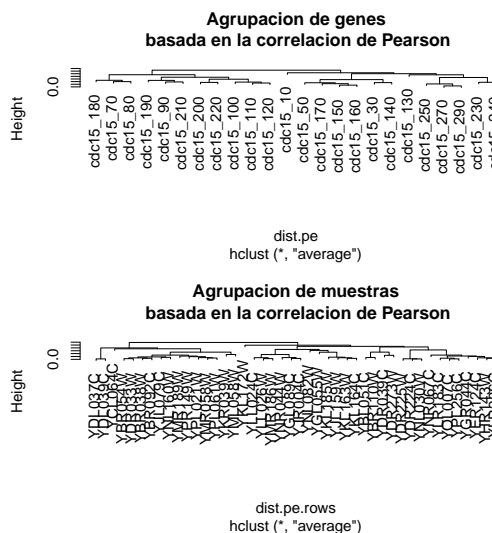
Visualización un dendrograma a partir de las correlación entre genes

A pesar que en los ejemplos anteriores se han utilizado distancias euclídeas, los perfiles de expresión génica acostumbran a agruparse en función de los coeficientes de correlación.

Para ello es necesario calcular la correlación entre los genes y cambiar el formato de la matriz de correlación para pueda contener las distancias. A continuación, el árbol puede dibujarse normalmente.

```
> cor.pe <- cor(as.matrix(dat.sel), method=c("pearson"))
> cor.pe.rows <- cor(t(as.matrix(dat.sel)), method=c("pearson"))
> cor.sp <- cor(as.matrix(dat.sel), method=c("spearman"))
> dist.pe <- as.dist(1-cor.pe)
> dist.pe.rows <- as.dist(1-cor.pe.rows)
> dist.sp <- as.dist(1-cor.sp)
> hc.cor <- hclust(dist.pe, method=treemeth)
> hc.cor.rows <- hclust(dist.pe.rows, method=treemeth)

> oldpar <- par(mfrow=c(2,1))
> plot(hc.cor, main="Agrupacion de genes\n basada en la correlacion de Pearson")
> plot(hc.cor.rows, main="Agrupacion de muestras\n basada en la correlacion de Pearson")
> par(oldpar)
```



11.1.3 Agrupación de genes por el método de las k-means

Si en lugar de muestras deseamos agrupar por genes es mejor usar métodos no jerárquicos partitivos como **k-means**.

Un inconveniente de l'agrupación por **k-means** es que hace falta escoger un número el número de clusters (k) antes de realizar la agrupación.

Por ejemplo para producir un agrupamiento k-means con 5 grupos haremos:

```
> k <- c(5)
> km <- kmeans(dat.sel, k, iter.max=1000)
> #summary(km)
> #head(km)
```

El método de las **k-means** permite estudiar la consistencia de la agrupación a partir del cálculo de las sumas de cuadrados dentro de cada grupo (*Within SS*). El promedio de estas sumas de cuadrados para todos los grupos creados (variable “withinss”) es una medida de la variabilidad dentro de los grupos, es decir qué tan parecidos son los genes dentro de los grupos.

```
> mean(km$withinss)
```

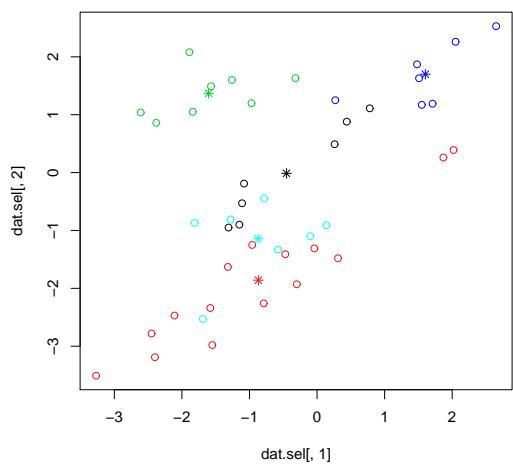
Una manera de evaluar las agrupaciones generadas mediante “k-means” es ejecutar el mismo análisis varias veces – guardando cada vez el resultado como un nuevo objeto – y seleccionar aquella agrupación que ofrece un menor valor de “withinss”.

Dibujo de una agrupación **k-means**

Los resultados de los métodos divisivos como **k-means** no pueden visualizarse mediante un dendrograma por lo que es preciso recurrir a otras representaciones gráficas.

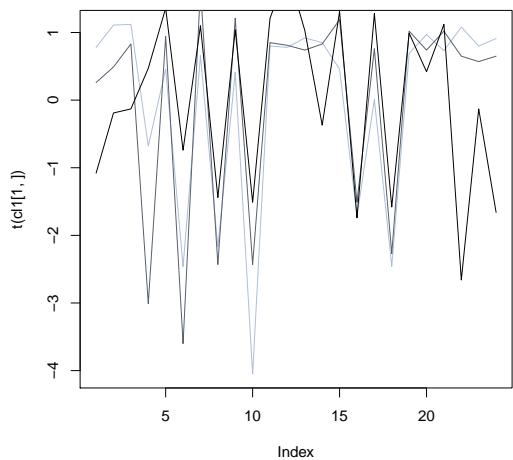
Podemos, por ejemplo, pintar los miembros de cada grupo con un color y un símbolo distintos.

```
> c1 <- km$cluster
> plot(dat.sel[,1], dat.sel[,2], col=c1)
> points(km$centers, col = 1:5, pch = 8)
```



Para visualizar un sólo grupo, y sus valores de expresión, seleccionaremos los genes que lo constituyen y a continuación, dibujaremos los perfiles de expresión de estos genes utilizando diferentes colores.

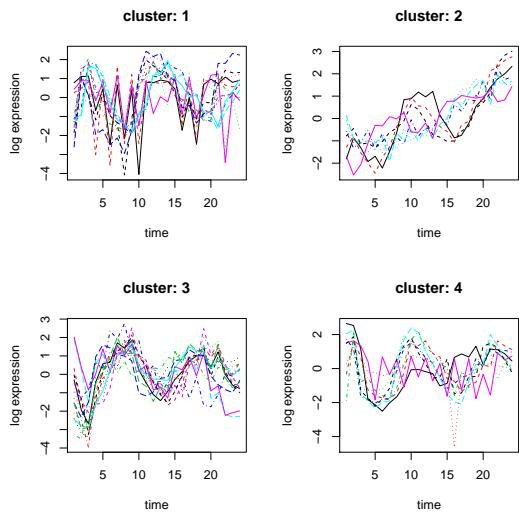
```
> set3 <- data.frame(dat.sel, cl)
> cl1 <- dat.sel[which(cl==1),] # Here we have 3 genes
> plotcol <- colorRampPalette(c("Grey", "Black"))(5)
> plot(t(cl1[,1]), type="l", col=plotcol[1])
> lines(t(cl1[,2]), type="l", col=plotcol[3])
> lines(t(cl1[,3]), type="l", col=plotcol[5])
```



Si hay más de un grupo podemos visualizarlos todos en el mismo gráfico utilizando un pequeño bucle **for**

```
> km <- kmeans(dat.sel, 4, iter.max=1000)
```

```
> par(mfrow=c(2,2))
> for(i in 1:4) {
+   matplot(t(dat.sel[km$cluster==i,]), type="l",
+   main=paste("cluster:", i), ylab="log expression", xlab="time")
+ }
```



11.1.4 Anotación de resultados

Una vez que se han identificado algunos grupos de interés, se puede comprobar que genes constituyen de cada uno de los “clusters”. Esto puede hacerse con el paquete de anotaciones para levadura de donde podremos extraer información sobre nombres, descripciones y otras anotaciones en múltiples bases de datos:

```
> if(!require(org.Sc.sgd.db)){
+   source("http://bioconductor.org/biocLite.R")
+   biocLite("org.Sc.sgd.db")
+ }
> require("org.Sc.sgd.db")
> anotData <- capture.output(org.Sc.sgd())
> print(anotData[1:15])

[1] "Quality control information for org.Sc.sgd:"
[2] ""
[3] ""
[4] "This package has the following mappings:"
[5] ""
[6] "org.Sc.sgdALIAS has 2559 mapped keys (of 8702 keys)"
[7] "org.Sc.sgdCHR has 8702 mapped keys (of 8702 keys)"
[8] "org.Sc.sgdCHRENGTHS has 17 mapped keys (of 17 keys)"
```

```
[9] "org.Sc.sgdCHRL0C has 8072 mapped keys (of 8702 keys)"  
[10] "org.Sc.sgdCHRLOCEND has 8072 mapped keys (of 8702 keys)"  
[11] "org.Sc.sgdCOMMON20RF has 9565 mapped keys (of 9565 keys)"  
[12] "org.Sc.sgdDESCRIPTION has 8348 mapped keys (of 8702 keys)"  
[13] "org.Sc.sgdENSEMBL has 5881 mapped keys (of 8702 keys)"  
[14] "org.Sc.sgdENSEMBL20RF has 5888 mapped keys (of 5888 keys)"  
[15] "org.Sc.sgdENSEMBLPROT has 5881 mapped keys (of 8702 keys)"
```

```
> cat ("... output continues until ", length(anotData), " lines.\n")
```

```
... output continues until 47 lines.
```

Resumen

El análisis de datos de microarrays es una disciplina que combina la bioinformática la estadística y la biología para esclarecer problemas que aparecen en el estudio de la expresión génica con microarrays, que son herramientas que permiten el estudio de la expresión de manera simultánea en todos los genes de un organismo. Con los microarrays se pueden tratar multitud de problemas entre los que podemos destacar la *comparación de clases*, el *descubrimiento de nuevos grupos* o la construcción de predictores.

%beginpreguntas

Bibliography

- [1] A. Alizadeh, M.B. Eisen, E. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, X. Yu, J.I. Powell, L. Yang, G.E. Marti, J. Hudson Jr, L. Lu, D.B. Lewis, R. Tibshirani, G. Sherlock, W.C. Chan, T.C. Greiner, D.D. Weisenburger, J.O. Armitage, R. Warnke, R. Levy, W. Wilson, M.R. Grever, J.C. Byrd, D. Botstein, P.O. Brown, and L.M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, February 2000.
- [2] David B Allison, Xiangqin Cui, Grier P Page, and Mahyar Sabripour. Microarray data analysis: from disarray to consolidation and consensus. *Nat Rev Genet*, 7(1):55–65, January 2006.
- [3] Alain Barrier, Pierre-Yves Boelle, Antoinette Lemoine, Antoine Flahault, Sandrine Dudoit, and Michel Huguier. [gene expression profiling in colon cancer]. *Bull Acad Natl Med*, 191(6):1091–101; discussion 1102–3, June 2007.
- [4] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57:289–300, 1995.
- [5] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.
- [6] M.J. Callow, S. Dudoit, E.L. Gong, T.P. Speed, and E.M. Rubin. Microarray Expression Profiling Identifies Genes with Altered Expression in HDL-Deficient Mice. *Genome Research*, 2000.
- [7] John M. Chambers. *Programming with data: a guide to the S language*. Springer, 1998.
- [8] R.L. Chelvarajan, Y. Liu, D. Popa, M.L. Getchell, T.V. Getchell, A.J. Stromberg, and S. Bondada. Molecular basis of age-associated cytokine dysregulation in LPS-stimulated macrophages. *Journal of Leukocyte Biology*, 79(6):1314, 2006.
- [9] MAQC Consortium. The MicroArray quality control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models. *Nat Biotech*, 28(8):827–838, 2010.
- [10] Carles M. Cuadras. *Análisis Multivariante*. EUNIBAR, 1989.
- [11] R. Diá. Supervised Methods with Genomic Data: a Review and Cautionary View. *Data analysis and visualization in genomics and proteomics*. New York: Wiley, pages 193–214, 2005.
- [12] R. Duda, P. Hart, and DG. Stork. *Pattern recognition*, 2nd. Ed. John Wiley and Sons, 2001.

- [13]S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):1–21, 2002.
- [14]S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457), 2002.
- [15]S. Dudoit, J. P. Shaffer, and J. C. Boldrick. Multiple hypothesis testing in microarray experiments. *Statistical Science*, 18:71–103, 2003.
- [16]A. Dupuy and R.M. Simon. Critical Review of Published Microarray Studies for Cancer Outcome and Guidelines on Statistical Analysis and Reporting. *JNCI Journal of the National Cancer Institute*, 99(2):147, 2007.
- [17]L. Dyrskjøt, T. Thykjaer, M. Kruhøffer, J.L. Jensen, N. Marcussen, S. Hamilton-Dutoit, H. Wolf, and T.F. Ørntoft. Identifying distinct classes of bladder carcinoma using microarrays. *Nature Genetics*, 33:90–96, 2002.
- [18]M. B. Eisen, P. T. Spellman, P. O. Brownand, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences USA*, 95(25):14863–14868, 1998.
- [19]Julian J. Faraway. *Linear Models with R*. Chapman and Hall/CRC, 1 edition, July 2004.
- [20]Simon Frantz. An array of problems. *Nat Rev Drug Discov*, 4(5):362–363, May 2005.
- [21]Y. Freund and R. Schapire. Experiments with a new boosting algorithm, in Machine Learning: Proceedings of the Thirteenth International Conference. *Morgan Kauffman, San Francisco*, pages 148–156, 1996.
- [22]Robert Gentleman, Vince Carey, Wolfgang Huber, Rafael Irizarry, and Sandrine Dudoit. *Bioinformatics and Computational Biology Solutions using R and Bioconductor*. Springer, New York, 2005.
- [23]D.H. Geschwind and J.P. Gregg. *Microarrays for the neurosciences: an essential guide*. MIT Press, 2002.
- [24]T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M.L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
- [25]T. Hastie, R. Tibshirani, and G. Walther. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society, B*, 63(41):1–423, 2001.
- [26]I. A. Hedenfalk, M. Ringnér, J. M. Trent, and A. Borg. Gene expression in inherited breast cancer. *Adv Cancer Res*, 84:1–34, 2002.
- [27]Sandrine Imbeaud and Charles Auffray. 'The 39 steps' in gene expression profiling: critical issues and proposed best practices for microarray experiments. *Drug Discovery Today*, 10(17):1175–1182, September 2005. PMID: 16182210.

- [28]R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, K. J. Antonellis, U. Scherf, and T. P. Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4:249–264, 2003.
- [29]L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. John Wiley and Sons, 1990.
- [30]M K Kerr and G A Churchill. Experimental design for gene expression microarrays., June 2001.
- [31]M.K. Kerr and G.A. Churchill. Bootstrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. *Proceedings of the National Academy of Sciences*, page 161273698, 2001.
- [32]P. Khatri and S. Drăghici. Ontological analysis of gene expression data: current tools, limitations, and problems. *Bioinformatics*, 18:3587–3595, 2005.
- [33]Isabelle Lesur Kupin. *Study of the Transcriptome of the prematurely aging dna-2 yeast mutant using a new system allowing comparative DNA microarray analysis*. PhD thesis, Universite Bordeaux I, April 2005.
- [34]P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J.A. Lozano, R. Armañanzas, G. Santafé, A. Perez, and V. Robles. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1):86–112, 2006.
- [35]Heidi Ledford. The death of microarrays? *Nature News*, 455(7215):847–847, October 2008.
- [36]M.L.T. Lee and GA Whitmore. Power and sample size for DNA microarray studies. *Statistics in Medicine*, 21(23):3543–3570, 2002.
- [37]Simon M Lin, Jyothi Devakumar, and Warren A Kibbe. Improved prediction of treatment response using microarrays and existing biological knowledge. *Pharmacogenomics*, 7(3):495–501, April 2006. PMID: 16610959.
- [38]G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.
- [39]Shanna Moore, Julia Vrebalov, Paxton Payton, and Jim Giovannoni. Use of genomics tools to isolate key ripening genes and analyse fruit maturation in tomato. *Journal of Experimental Botany*, 53(377):2023–2030, 2002.
- [40]J-L. Mosquera and A. Sánchez-Pla. A comparative study of go mining programs. In *X Conferencia Española de Biometría*. Sociedad Española de Biometría, 2005.
- [41]J. Quackenbush. Microarray data normalization and transformation. *Nature Genet.*, 32:496–501, 2002.
- [42]P. Rousseeuw, E. Trauwaert, and L. Kaufman. Some silhouette-based graphics for clustering interpretation. *Belgian Journal of Operations Research, Statistics and Computer Science*, 29(3):35–55, 1989.

- [43]A. Sánchez-Pla and J.L Mosquera. The quest for biological significance. In L.L. Bonilla, M. Moscoso, G. Platero, and J.M. Vega, editors, *Progress in Industrial Mathematics at ECMI 2006*. Springer, New York, 2007.
- [44]M. Schena. *Microarray Analysis*. J. Wiley & Sons, Hoboken, NJ, 1999.
- [45]Denise Scholtens, Alexander Miron, Faisal M. Merchant, Arden Miller, Penelope L. Miron, J. Dirk Iglehart, and Robert Gentleman. Analyzing factorial designed microarray experiments. *J. Multivar. Anal.*, 90(1):19–43, 2004.
- [46]Richard M. Simon, Edward L. Korn, Lisa M. McShane, Michael D. Radmacher, George W. Wright, and Yingdong Zhao. *Design and Analysis of DNA Microarray Investigations*. Springer-Verlag, 2003.
- [47]Gordon K Smyth, Joëlle Michaud, and Hamish S Scott. Use of within-array replicate spots for assessing differential expression in microarray experiments. *Bioinformatics*, 21(9):2067–75, May 2005.
- [48]T. Speed. *Statistical Analysis of Gene Expression Data*. Boca Raton, Fla.: Chapman & Hall/CRC, 2003.
- [49]R. Tibshirani. A simple method for assessing sample sizes in microarray experiments. *BMC Bioinformatics*, 7(1):106, 2006.
- [50]Laura J van ’t Veer, Hongyue Dai, Marc J van de Vijver, Yudong D He, Augustinus A M Hart, Mao Mao, Hans L Peterse, Karin van der Kooy, Matthew J Marton, Anke T Witteveen, George J Schreiber, Ron M Kerkhoven, Chris Roberts, Peter S Linsley, René Bernards, and Stephen H Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–6, January 2002.
- [51]Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63, January 2009.
- [52]J. Won Lee, J. Bok Lee, M. Park, and S. Song. An extensive comparison of recent classification tools applied to microarray data. *Computational Statistics & Data Analysis*, 48(4):869–885, 2005.
- [53]Y. H. Yang, M. J. Buckley, S. Dudoit, and T. P. Speed. Comparison of methods for image analysis on cDNA microarray data. *Journal of Computational and Graphical Statistic*, 11(1), 2002.
- [54]KY Yeung, C. Fraley, A. Murua, AE Raftery, and WL Ruzzo. Model-based clustering and data transformations for gene expression data, 2001.
- [55]Qianqian Zhu, Jeffrey Miecznikowski, and Marc Halfon. Preferred analysis methods for affymetrix genechips. ii. an expanded, balanced, wholly-defined spike-in dataset. *BMC Bioinformatics*, 11(1):285, 2010.