UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

genArise: herramienta de software para el análisis de microarreglos

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN Y LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A N:

ANA PATRICIA GÓMEZ MAYÉN
GUSTAVO CORRAL GUILLÉ

DIRECTOR DE TESIS:

BIOL. GERARDO COELLO COUTIÑO

CODIRECTOR DE TESIS:

DR. PEDRO MIRAMONTES VIDAL

FACULTAD DE CIENCIAS
UNAM

2005

VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

genArise: herramienta de software para el análisis de microarreglos.

realizado por **GÓMEZ MAYÉN ANA PATRICIA** y **CORRAL GUILLÉ GUSTAVO**, con número de cuenta **09719676-3** y **09710639-5** respectivamente, quienes cubrieron los créditos de la carrera de **LIC. EN CIENCIAS DE LA COMPUTACIÓN**.

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director
Propietario **Biol. Gerardo Coello Coutiño**

Codirector
Propietario **Dr. Pedro Eduardo Miramontes Vidal**

Propietario **Dra. Lina Riego Ruiz**

Suplente **Dr. Humberto Andrés Carrillo Calvet**

Suplente **Dra. Hanna Oktaba**

Consejo Departamental de Matemáticas

Dr. Francisco Hernández Quiroz
Coordinador de la Carrera de
Ciencias de la Computación

ACT. MAURICIO AGUILAR GONZÁLEZ
CONSEJO DEPARTAMENTAL
DE
MATEMATICAS

A mi mamá María Antelma Mayén Galicia.
A mi papá Hugo Salvador Gómez Garay
A mis hermanos Diana y Hugo
Al amor de mi vida Egar...
Paty

A mis padres por su apoyo.
A Ismene por su paciencia.
A la banda.
Gustavo

Agradecimientos

Queremos agradecer a todos aquellos que han aportado algo a lo largo de nuestra carrera y en especial a lo largo del proyecto de esta tesis. Al Biol. Gerardo Coello y a la Dra. Lina Riego por todo su apoyo para la elaboración de este trabajo. Al Dr. Jorge Ramírez por su ayuda en la difusión del sistema. En especial a Sergio Rojas, ya que no sólo nos avisó de la oportunidad de realizar esta tarea sino también nos ha brindado su amistad en todo momento.

Hay que agradecer la ayuda de nuestros amigos con los que sufrimos y disfrutamos gran parte de la carrera: Laura Leonides, Emmanuel Plata, Rodrigo Poblanno, Yazmín Ibáñez y Didier Gutiérrez.

A todos, muchas gracias.

Índice general

Índice de figuras

Índice de cuadros

Introducción

El ADN de un organismo está compuesto en gran parte por genes y algunos otros elementos y secuencias de las que no se conoce mucho. Dentro de cada célula estos genes se activan o desactivan de acuerdo a la función que ocupa la célula en un tejido en particular. Se dice que los genes que se encuentran activos están expresados, y los niveles de expresión pueden variar como respuesta a estímulos ajenos al genoma como la temperatura, el alimento y los rayos solares. De la misma manera, los organismos responden a través de la expresión de su genoma ante el contacto con fármacos, infecciones o diferentes estados patológicos, es por ello que las herramientas que permiten estudiar los niveles de expresión genómica de un organismo, hacen posible conocer algunos padecimientos a los que éste es susceptible, con la ambición de diseñar tratamientos específicos que modifiquen nuevamente las expresiones alteradas.

Los microarreglos de ADN son una de las tecnologías más recientes para el estudio de la expresión global de genes. Esta técnica muestra, en un solo experimento, los genes que están siendo activados y expresados en forma de ARNm, en un grupo celular determinado y bajo estímulos especifícficos de cada experimento.

En el campo clínico, los microarreglos han facilitado la comprensión de los mecanismos moleculares de carcinogénesis, la identificación de nuevos marcadores tumorales, la clasificación de neoplasias de difícil diferenciación, así como la determinación de los genes involucrados en las respuestas inmunes en diferentes enfermedades o en la respuesta a diferentes microorganismos.

Antes de poder obtener los genes diferencialmente expresados, bajo las condiciones determinadas, y las posibles relaciones entre ellos, es necesario aplicar un preprocesamiento a los datos de entrada obtenidos de los valores de un microarreglo. Este paso ayuda a compensar posibles variaciones y errores que se presentan en el transcurso del experimento, ajustando las medidas de las intensidades del arreglo, además de eliminar información redundante o que se juzgue de poco interés para el análisis.

En la actualidad existen varias herramientas, tanto comerciales como libres, para realizar el análisis de microarreglos; sin embargo, en ambos casos, el algoritmo empleado en algunas operaciones del procesamiento puede no ser el que el investigador desea aplicar a sus datos. Esta es la razón principal por la que algunos investigadores optan por usar más de un programa para realizar un sólo análisis, con los datos obtenidos de un microarreglo, con el objetivo de obtener los resultados esperados. Además no existe un formato de entrada estándar entre las diferentes aplicaciones, lo que obliga al investigador a modificar manualmente el formato de sus datos para poder utilizar parcialmente diferentes herramientas en cada paso del análisis que así lo requiera.

El objetivo de este proyecto fué desarrollar una herramienta de software en la cual estuvieran integrados diferentes algoritmos para realizar el preprocesamiento de los datos provenientes de un microarreglo, esto, con la finalidad de proporcionarle al usuario la oportunidad de seleccionar qué transformación aplicar a los datos durante la fase de preprocesamiento y el orden en las que éstas son aplicadas brindando una mayor flexibilidad en el desarrollo del análisis. El resultado de este proyecto es una aplicación que conjunta estos requerimientos a la que hemos denominado *genArise*.

Dado que la fiabilidad de los resultados del análisis depende en gran medida de una correcta implementación de las operaciones estadísticas, de las que hacen uso los algoritmos del preprocesamiento y análisis de los datos, genArise está completamente desarrollado en R [22].

R es un ambiente interactivo para el análisis de datos, cómputo estadístico y visualización, distribuido de manera libre y que además cuenta con un lenguaje que lo hace programable, capaz de aplicarse en la resolución de problemas complejos, así como en el análisis de datos.

Gracias a las características del lenguaje R, genArise proporciona a los usuarios una interfaz intuitiva en donde pueden diseñar sus protocolos de análisis, ejecutando una o más operaciones para el preprocesamiento de los datos. Además fué posible la implementación de una GUI (Interfaz Gráfica de Usuario) que le permitiera al usuario la visualización de los datos después de cada transformación realizada.

Aunque inicialmete genArise fue una herramienta diseñada para analizar datos generados en la Unidad de Microarreglos del Instituto de Fisiología Celular de la UNAM y para los investigadores de este instituto, durante el desarrollo de este proyecto se le han hecho modificaciones para que pueda aceptar datos de microarreglos obtenidos fuera de la Unidad.

Tras una fase de pruebas, se tomó la decisión de compartir genArise por medio

de *Bioconductor*, un sitio especializado en herramientas para la bioinformática implementadas en R [1], incluyendo una completa documentación de las funciones, así como una guía de utilización para el usuario. De esta manera genArise se encuentra ahora disponible como software libre y está en constante mantenimiento.

En esta tesis se exponen los fundamentos de genArise: sus alcances y limitaciones, una descripción del lenguaje utilizado, así como las conclusiones y planes a futuro surgidos a lo largo de este proyecto.

En el capítulo primero se exponen los puntos medulares de la genética y la genómica, el rápido desarrollo que esta área ha experimentado en años recientes, siendo el proyecto del genoma humano un claro ejemplo, resaltando esfuerzos realizados y los avances de la genética y la genómica en México.

En el capítulo segundo se desarrolla el tema de los microarreglos de ADN describiendo el protocolo que se sigue durante su elaboración y algunas de las tecnologías utilizadas durante esta etapa. La descripción y justificación de cada una de las operaciones de transformación que se aplican a los datos para su posterior análisis se describen en el capítulo tercero.

En el capítulo cuarto nos enfocamos de lleno al funcionamiento de genArise además de describir cada uno de los módulos que esta aplicación contiene y el desarrollo que se llevó a cabo. Se presenta también una breve justificación del lenguaje empleado para la implementación y del diseño del software. Este es, en todo caso, el capítulo donde está plasmado el trabajo realizado para este proyecto.

Finalmente en el capítulo quinto pretendemos presentar de la manera más clara y concisa las especificaciones del lenguaje R, incluyendo aspectos de diseño como la sintaxis, las reglas para la evaluación de expresiones, evaluación de los argumentos de las funciones, sistema de tipos y otras características más particulares como la interfaz con C y Fortran, la implementación de interfáz gráfica en R, entre otras.

[1] `http://www.bioconductor.org`

Capítulo 1

Genes y genómica

"Lo esencial es invisible para los ojos..."
El Principito, Antoine De Saint-Exupéry

La *célula* es la unidad básica de vida, independientemente de su estructura y organización interna, en ella está contenido el material genético, el cual posee la información explícita y única de cada célula. El material genético es un componente químico que se conoce como *ácido desoxirribonucleico (ADN)*, el *genoma* de un organismo está compuesto de todo el ADN contenido en sus células.

Conocer con precisión la información inscrita en el genoma de un organismo es el primer paso para realizar una serie de investigaciones a partir del conocimiento de la secuencia, ya que al descifrar la información que el genoma contiene, puede aportar datos relevantes y específicos de cómo reacciona el organismo ante factores externos. En el ser humano, por ejemplo, se pueden predecir aquellas enfermedades a las que un individuo es susceptible y, con ello, se podría tratar de evitar o retrasar algunos padecimientos relacionados con la enfermedad, sus complicaciones y secuelas, siempre y cuando se sepa a ciencia cierta qué fué lo que originó el padecimiento.

Sin embargo, a pesar de los beneficios que brinda el estudio del genoma humano, existe temor en algunos sectores de la población sobre el uso que se le puede dar a la información obtenida, ya que se puede conocer información propia y única de cada individuo en particular. Esto ha generado una discusión a nivel mundial y se ha coincidido en la necesidad de una legislación para regular el uso y la obtención de la información relacionada con el genoma humano.

El genoma humano se ha considerado patrimonio de la humanidad [1], por lo que los resultados que arrojen las investigaciones en torno a éste deben responder a diferentes sectores de la población, uno de ellos: la comunidad científica. Es necesario que se respete el derecho a la investigación en torno al genoma humano, pero al tratarse de

una información que es patrimonio de la humanidad, se debe garantizar que las investigaciones no tienen únicamente fines lucrativos. También es necesario que se respeten los derechos humanos en todo momento, incluyendo en el manejo que se le puede dar a la información obtenida. De la misma manera se debe respetar la ideología en ciertas regiones del mundo, repecto a todo lo relacionado al genoma humano. El debate sobre la equidad en la obtención y el uso de la información en torno al genoma humano y sus repercusiones no ha tenido avances tan rápidos como su investigación.

1.1. La célula y el ADN

Todos los organismos vivientes están compuestos por células, de hecho se acepta que ningún organismo es un ser vivo si no consta de al menos una célula. A pesar de que los virus y los componentes de la célula, conocidos como *organelos*, realizan muchas de las funciones propias de la célula viva, carecen de la capacidad de crecimiento y reproducción propios de la célula.

A pesar de las diferencias fundamentales que existen entre las células (tamaño, forma y estructura) todas contienen información hereditaria codificada en las moléculas de *ADN*, las cuales dirigen la actividad de la célula y garantizan la reproducción y el paso de caracteres a la descendencia. En las *células eucariotas* el ADN se encuentra empaquetado junto con proteínas en unas estructuras llamadas *cromosomas*(Figura 1.1), las cuales se localizan dentro del *núcleo*; sin embargo, el ADN también se encuentra presente en organelos como las *mitocondrias* y los *cloroplastos*(plantas) que se localizan en el *citoplasma*.

En 1953 se presentó la estructura de la molécula de ADN por el biofísico británico Francis Harry Compton Crick y el bioquímico estadounidense James Dewey Watson, quienes al estudiar fotografías del ADN con rayos X, tomadas por Rosalyn Franklin, dedujeron la estructura química real de esta molécula, trabajo con el cual obtuvieron, junto con Maurice Wilkins, jefe de Franklin, el premio Nobel de Medicina en 1962.

Según el modelo de Watson y Crick, el ADN es una molécula constituida por dos cadenas o bandas formadas por un elevado número de compuestos químicos llamados *nucleótidos*. Cada nucleótido está formado por tres unidades: una molécula de azúcar llamada *desoxirribosa*, una molécula de ácido fosfórico conocido como *fosfato* y uno de cuatro posibles compuestos nitrogenados conocidos como *bases nitrogenadas*. Las bases nitrogenadas en el ADN pueden ser adenina (A), timina (T), guanina (G) y citosina (C). Debido a su afinidad química, la adenina generalmente se une a la timina y la guanina generalmente se une a la citosina (Figura 1.2).

La molécula de desoxirribosa es el centro del nucleótido y esta flanqueada por un grupo fosfato de un lado y una base nitrogenada al otro. El grupo fosfato está unido a su vez a la desoxirribosa del nucleótido adyacente formando una cadena. Estas sub-unidades enlazadas desoxirribosa-fosfato forman los postes de una escalera, los

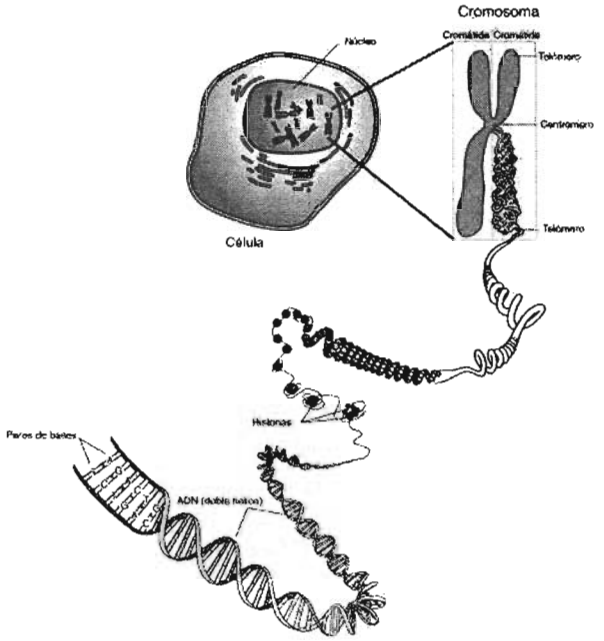

Figura 1.1: Los cromosomas en una célula eucariote. Tomado de [2]

travesaños están formados por las bases complementarias que se unen entre sí por enlaces químicos llamados *enlaces de hidrógeno*. La escalera formada se enrolla sobre un eje central dándole a la cadena de ADN la estructura de *doble hélice* (Figura 1.3).

1.1.1. Los genes

Un *gen* es un fragmento de ADN formado por una secuencia de nucleótidos que puede contener desde 100 pares de bases hasta dos millones de pares de bases [2]. En muchas ocasiones se dice que un gen es una secuencia de nucleótidos que generalmente codifica para una proteína.

Para diferenciar cada uno de los genes a lo largo de la cadena de ADN, cada gen está dividido en tres partes: promotor, región codificante y secuencia terminal. El promotor y la secuencia terminal son secuencias de nucleótidos que determinan en dónde comienza y termina un gen.

La región codificante se encuentra entre el promotor y la secuencia terminal, pero puede estar interrumpida por segmentos que no contienen información para la síntesis de alguna proteína, es decir, son secuencias que no codifican pero que inicialmente

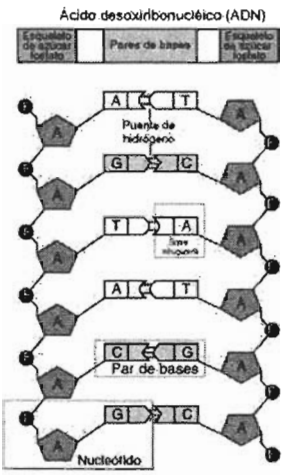

Figura 1.2: Nucleótidos del ADN. Tomado de [2]

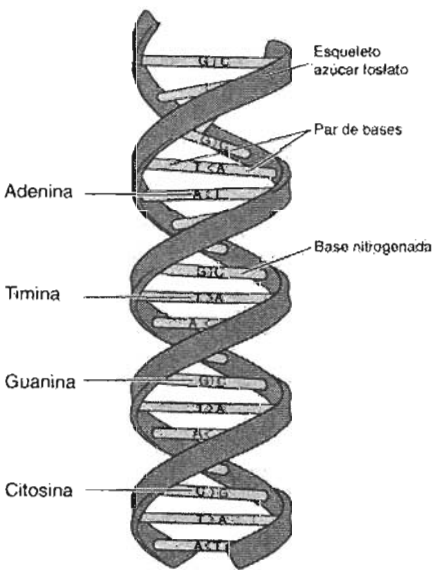

Figura 1.3: Estructura del ADN. Tomado de [2]

se copian a la molécula de ARNm durante la transcripción, el cual se detallará más

adelante.

Los segmentos que interrumpen la región codificante se denominan *intrones* y los segmentos que forman la secuencia codificante se denominan *exones* (Figura 1.4). Un gen puede contener varios exones que están alternados con intrones. Las funciones de los intrones, si existen, tadavía se desconocen.

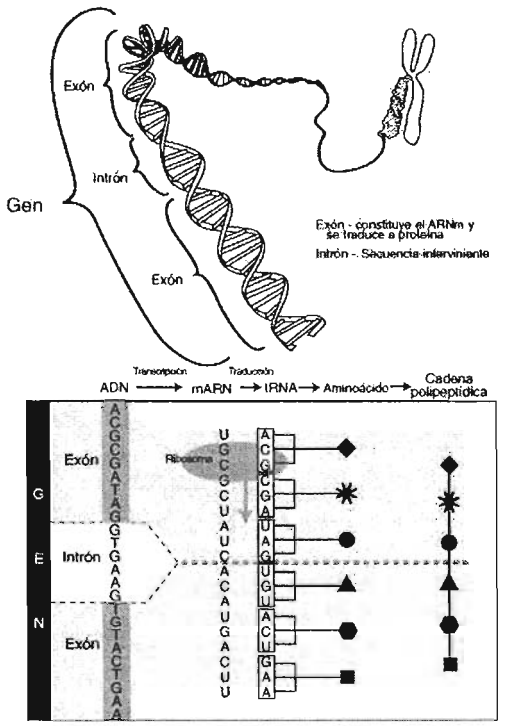

Figura 1.4: Flujo de la información genética. Tomado de [2]

Debido a la presencia de los intrones en los genes, identificar qué genes pertenecen al genoma de un organismo no es cosa sencilla, una vez obtenida la secuencia de bases que pertenecen a la cadena de ADN del organismo en cuestión, se deben obtener únicamente las secuencias codificantes; sin embargo, identificar los genes que pertenecen al genoma de un organismo en particular es la parte fácil, ya que con la lista de los genes no se puede hacer mucho. La información del genoma se vuelve valiosa cuando podemos correlacionar a un gen con su función[1].

[1] Hasta ahora no se sabe cuál es la función del 40 % de los genes que se han identificado en la secuencia del genoma humano [3]

1.1.2. La síntesis de proteínas

Las *proteínas* son unas moléculas que desempeñan una amplia gama de actividades en la célula y es el ADN el que determina su estructura. En el ser humano, el cabello y los músculos están compuestos por proteínas al igual que las enzimas que nos ayudan a digerir los alimentos; cuando las proteínas tienen una estructura alterada pueden ocasionar enfermedades, lo mismo sucede cuando hay una deficiencia o una saturación de alguna de estas moléculas, simplemente el organismo no funciona correctamente.

Cada célula puede sintetizar varias proteínas y, aunque es el ADN el que posee la codificación para elaborarlas, se vale de otros ácidos nucleicos llamados *ácidos ribonucleicos (ARNs)* para poder llevar a cabo la síntesis de las proteínas. Al igual que el ADN, el ARN es una molécula compuesta por nucleótidos, pero las bases posibles que se pueden encontrar en los nucleótidos de ARN son: adenina (A), guanina(G), uracilo (U) y citosina (C). Estos compuestos se unen como en el ADN, la diferencia química es que la molécula de azúcar en el ARN contiene un átomo de oxígeno que no está en la molécula de ADN y se emplea una base de uracilo en lugar de timina (Figura 1.5).

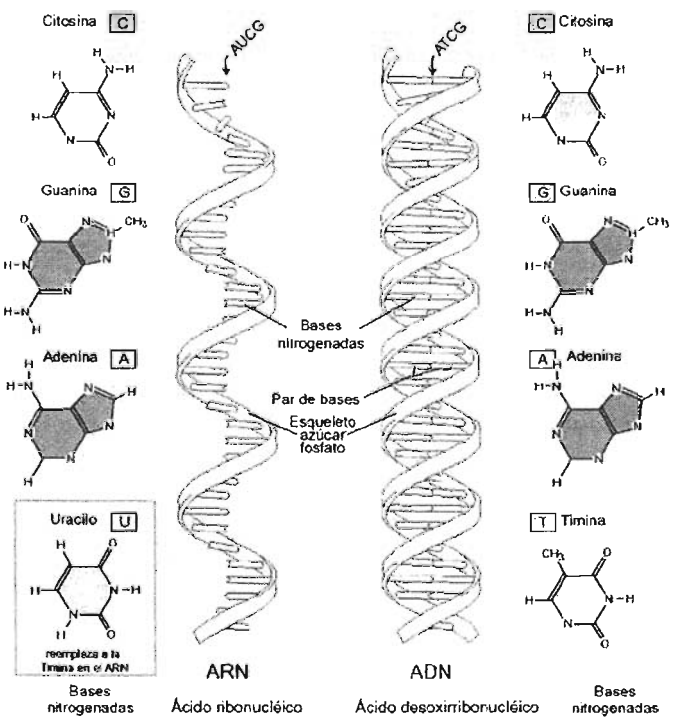

Figura 1.5: Estructura del ARN y ADN. Tomado de [2]

Hay tres tipos de ARN en las células eucariotas: *ARN ribosómico*, *ARN de transferencia* y *ARN mensajero*. Cada uno de estos ácidos nucleicos se van formando conforme son requeridos durante el proceso de síntesis proteica (Figura 1.6) y cada uno de ellos cumple una función importante para que este proceso pueda llevarse a cabo.

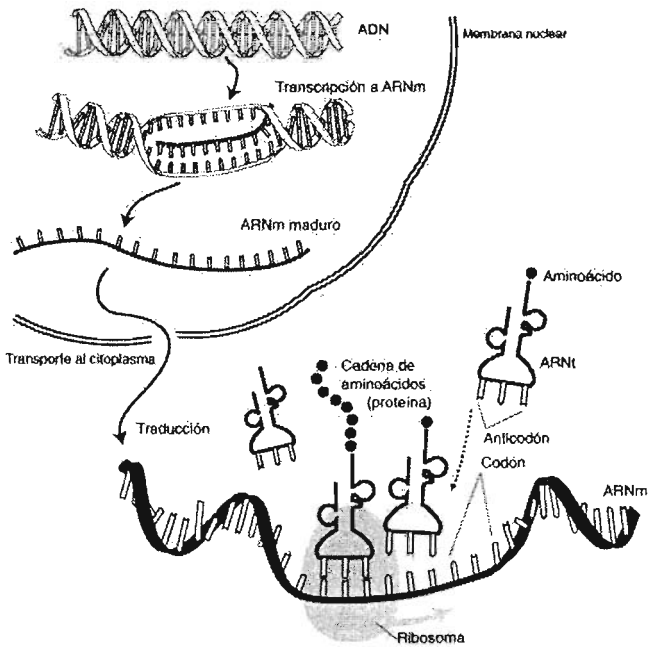

Figura 1.6: El proceso de síntesis de proteínas. Tomado de [2]

En el proceso de síntesis proteica también participan unas estructuras celulares llamadas *ribosomas* que se encuentran en el citoplasma, estos ribosomas actúan como centro de *síntesis proteica* y se requiere de una molécula intermediaria que copie la información del ADN y las traslade a los ribosomas, esta molécula intermediaria se llama *ARN mensajero*.

En las células eucariotas el ARN mensajero saca la información del núcleo y la traslada hacia el citoplasma en conde se encuentran los ribosomas. En el caso de las bacterias no hay compartimientos, los ribosomas y el ADN se encuentran en el citoplasma, pero también hay una molécula de ARN mensajero que copia la información inscrita en el ADN.

Durante el proceso de *transcripción* se hace una copia complementaria de un fragmento de ADN, el cual contiene la información que codifica una instrucción en particular. En una primera etapa, una enzima conocida como *ARN polimerasa* se asocia a una región específica del ADN llamada *promotor*. La enzima ARN polimerasa separa

las dos hebras de la molécula de ADN dando una vuelta a la hélice y permitiendo la polimerización del ARN a partir de una de las hebras de ADN que se utiliza como patrón. La enzima ARN polimerasa se desplaza a lo largo de la *cadena codificante* insertando nucleótidos de ARN de acuerdo a la complementariedad de las bases. La secuencia de nucleótidos que forma el promotor no es copiada, solo indica a partir de dónde se empieza a copiar el ARNm, de la misma manera, existe una secuencia de nucleótidos que indica hasta dónde se deja de copiar la información contenida en el ADN, esta secuencia es conocida como *secuencia terminal*. Una vez que la enzima ARN polimerasa ha encontrado la secuencia terminal se separa del ADN y permite que las dos hebras vuelvan a juntarse.

Antes de que termine el proceso de transcripción, en las células eucariotas, el ARNm comienza a desprenderse de la cadena codificante y se eliminan las secuencias de los fragmentos que no codifican (el ARN madura). Una vez que se ha terminado el proceso de transcripción en la célula eucariota, el ARNm sale del núcleo por unos poros que existen en la membrana nuclear y llega al citoplasma en donde se acopla a los ribosomas para realizar la síntesis de la proteína correspondiente.

El ARN de transferencia lleva *aminoácidos* a los ribosomas una vez que el ARNm haya llegado a éstos para incorporarlos a la proteína que será sintetizada. Una *proteína* es una molécula compuesta por una o más cadenas de aminoácidos y cada aminoácido es incorporado a la proteína por el reconocimiento de *codones*. En asociación con el ribosoma se forman enlaces químicos entre los aminoácidos, conocidos como *enlaces peptídicos*, el fenómeno en el que ésto sucede se llama *traducción*. Las cadenas pequeñas de aminoácidos unidos por enlaces peptídicos se definen como *péptidos* y las cadenas más largas como *polipéptidos*. Todas las proteínas son polipéptidos, pero no todos los polipéptidos son proteínas, por lo general, los polipéptidos con funciones específicas son proteínas.

La síntesis de proteínas es indispensable para la vida de las células; sin embargo, cada una de las proteínas se crean con la codificación de un fragmento de ADN (delimitado por el promotor y la secuencia terminal). Esto se debe a que las células fabrican las proteínas conforme se requieren y dependiendo de la función que ocupa la célula en un tejido en particular. Generalmente, los fragmentos de ADN que codifican una proteína en particular se denominan *genes*.

1.2. Expresión y regulación génica

La *expresión génica* es el proceso por medio del cual todos los organismos, ya sean procariotes o eucariotes, transforman la información codificada en el ADN de sus células en las proteínas necesarias para su desarrollo y funcionamiento. El conocimiento de los niveles de expresión génica de un organismo permite comprender como los genes producen efectos específicos sobre las estructuras y funciones de los organismos a los que pertenecen.

En todos los organismos, inclusive en los eucariotes, el contenido del ADN en todas sus células es el mismo y por ello, todas las células tienen la información necesaria para realizar la síntesis de todas las proteínas; pero no todos los genes se expresan al mismo tiempo ni en todas las células. Aunque existe un grupo de genes que se expresa en todas las células, el resto de los genes se expresan o no dependiendo de la función que realiza la célula a la que pertenecen en un tejido en particular. Por ejemplo, los genes que se encargan de la producción de proteínas necesarias para digerir los alimentos no están expresados en las neuronas o en las células que se encuentran en tejidos de la piel o músculos (Figura 1.7).

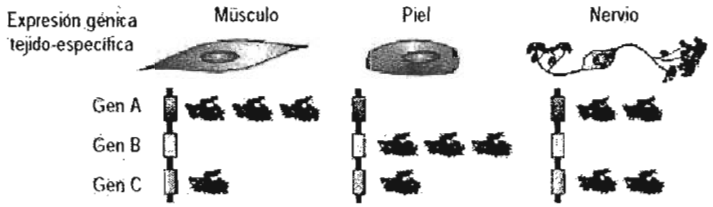

Figura 1.7: Esquema de los distintos niveles de expresión génica tejido-específica. Modificado de [4]

Cuando un gen se expresa se dice que está activo. A lo largo de la vida de un organismo existen conjuntos de genes que se activan, otros que se inactivan y algunos que simplemente cambian sus niveles de expresión, de acuerdo a una condición específica, pero se siguen expresando. Todas las células tienen mecanismos para regular la expresión de los genes según la función que codifiquen cada uno de ellos; esto se conoce como *regulación génica*. Mediante la regulación génica, las células sintetizan en cada momento sólo aquellos elementos que necesiten, indicando cuáles genes se activan o inactivan en un momento específico.

Las proteínas juegan un papel importante en el proceso de regulación génica. Todas las instrucciones de funcionamiento con las que cuenta un organismo se encuentran codificadas en su material genético. Las proteínas son las que ejecutan las acciones codificadas en los genes. Cada proteína puede presentar una acción diferente en la célula, algunas favorecen reacciones químicas, otras son elementos estructurales de la célula y, otras más, funcionan regulando el momento en el que se tienen que activar o inactivar ciertos grupos de genes.

En el caso de las bacterias, entre el promotor y la secuencia codificante, existe con frecuencia otro segmento de ADN conocido como *operador* en el cual se puede adherir alguna proteína que detiene el desplazamiento de la enzima ARN polimerasa y el proceso de transcripción, lo que impide que se produzca ARN mensajero; por lo tanto, el gen deja de expresarse.

Las proteínas que tienen la función de evitar el proceso de transcripción se llaman *represores*; sin embargo, aunque existan represores, la presencia en la célula de una

sustancia química determinada puede provocar que el represor se separe y el gen se active. Los genes que producen las proteínas represoras son conocidos como *genes reguladores*.

En las células eucariotas existen secuencias de ADN conocidas como *intensificadores* las cuales no forman parte de un gen en particular; sin embargo, si alguna proteína específica se une al intensificador, se provoca un plegamiento de la molécula de ADN, de tal forma que se puede acercar uno o varios genes a una región del cromosoma en donde se esté llevando a cabo el proceso de transcripción; esta acción activará dichos genes o por lo menos aumentará la velocidad de transcripción de los genes que se sitúen en el radio de acción del intensificador, los cuales, generalmente, se encuentran relacionados.

1.3. Mutaciones

Aunque no es muy común, se pueden producir errores cuando una célula se va a replicar a sí misma y el ADN que se copia a la célula hija es diferente al ADN contenido en la célula madre. Esta susitución hace que todas las células descendientes tengan la misma secuencia de bases alterada. Como resultado de la sustitución, los polipéptidos resultantes pueden ser distintos lo que puede originar que algunas funciones del organismo no se ejecuten correctamente. Esta alteración de la molécula de ADN se llama *mutación* y son cambios permanentes en el ADN.

En la mayoría de los casos, tales cambios pueden no tener algún efecto, o por el contrario pueden causar daño; sin embargo, en ocasiones una mutación puede mejorar la probabilidad de supervivencia de un individuo y pasar el cambio positivo a la descendencia [2].

La sustitución de algunos nucleótidos no es el único tipo posible de mutación, en algunas ocasiones se puede ganar (adición) o perder (deleción o ablación) por completo un nucleótido. Además es posible que se altere la forma y número de cromosomas, a este tipo de mutaciones se le conocen como *mutaciones cromosómicas* (Figura 1.8).

Existen mutaciones que no se manifiestan desde el nacimiento sino que se desarrollan en alguna etapa de la vida, otras mutaciones se adquieren y algunos factores como la dieta, el estilo de vida y el medio ambiente, propician su aparición, esta es la razón por la que muchas enfermedades hereditarias se presentan en alguna etapa de la vida. Cuando se presenta alguna mutación en uno o varios genes, puede que algunas funciones del organismo no se ejecuten correctamente.

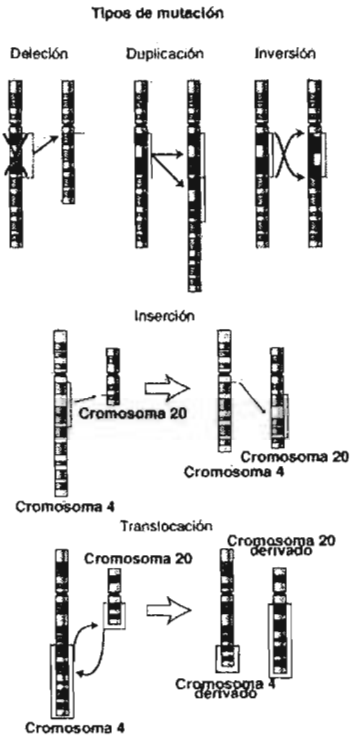

Figura 1.8: Algunos tipos de mutaciones cromosómicas. Tomado de [2]

1.4. El proyecto del genoma humano

En 1985 surge oficialmente la idea de secuenciar el genoma humano completo. En ese año se realizó una reunión en Santa Fe, Nuevo México para llevar a cabo la evaluación inicial de la factibilidad de la iniciativa. Esta evaluación estuvo a cargo de la Oficina de Investigación en Salud y Medio Ambiente del *Departamento de Energía de los Estados Unidos de América (DOE)*. Como resultado de la reunión, esa dependencia anunció "La Iniciativa del Genoma Humano"en 1986, asignando un presupuesto de 3.5 millones de dólares para el desarrollo de recursos estratégicos y proyectos piloto.

En 1987, el Congreso de Estados Unidos de América recomendó el desarrollo de un proyecto multidisciplinario con el fin de secuenciar y elaborar un mapa del genoma humano completo, lo que originó que el DOE estableciera los primeros centros multidisciplinarios del genoma humano. El Instituto Nacional de Salud de ése país (NIH) comenzó a financiar proyectos genómicos. A finales de ese año se fundó la *Organización Mundial del Genoma Humano* (HUGO) con el fin de coordinar esfuerzos a nivel internacional, además de establecer los planes de cooperación sobre el tema entre los

países participantes.

Se estableció un grupo de trabajo, formado por el DOE y el NIH en 1989, sobre los aspectos éticos, legales y sociales que pudieran surgir conforme se fuera obteniendo la información del genoma humano. Estas dependencias presentaron en 1990 un proyecto conjunto cuyo objetico principal sería la secuenciación de los 23 cromosomas humanos completos; fué así como se creó oficialmente, en ese año *El Proyecto del Genoma Humano*.

Poco después de su creación, el proyecto público pasó de ser un esfuezo limitado a ciertos grupos de Estados Unidos de América, a convertirse en un consorcio mundial, dirigido por el DOE y el NIH, en donde participaron diversos centros de secuenciación de ADN en países como Inglaterra, Francia, Japón, Alemania y China. La responsabilidad de los diferentes países involucrados en el PGH consistía en tomar la responsabilidad de clonar, mapear y secuenciar diferentes cromosomas humanos. Se definieron los lineamientos de acceso y uso común de los recursos del Proyecto del Genoma Humano.

Al mismo tiempo que se realizaban los trabajos del PGH, se pudo obtener la secuencia completa de los genomas de otros organismos, lo que permitió hacer algunas comparaciones entre las secuencias parcialmente obtenidas del genoma humano. En 1995, Hamilton Smith y Craig Venter lograron secuenciar el primer genoma completo de un organismo no viral: la bacteria *Hamophilus influenzae* [5]. Casi de manera simultánea se obtuvo la secuencia de la bacteria más pequeña: el *Mycobacterium genitalium* [6], con lo que se pudo obtener un modelo del mínimo número de genes necesarios para la existencia independiente de un organismo.

En 1996 se pudo obtener la secuencia de la levadura *Saccharomyces cerevisiae*, esta fue la primera secuencia obtenida de un organismo eucariote. En ese mismo año se establecieron las primeras leyes en Estados Unidos de América que prohíben el uso de la información genética en decisiones de acceso a seguros de gastos médicos [7].

En 1998, una compañía privada llamada *Celera Genomics* anunció su intención de secuenciar el genoma humano en un periodo de tres años, mediante el uso de estrategias diferentes a las utilizadas por el consorcio internacional y, obviamente, con fines comerciales.

Tras el riesgo de que una información tan valiosa pudiera ser patentada, se comenzó una carrera entre el proyecto financiado con fondos públicos y el proyecto privado por obtener la secuencia completa del genoma humano. En 1999 se logró terminar la secuencia del primer cromosoma humano completo: el cromosoma 22. En ese año se formó un consorcio para la identificación de variaciones en el genoma humano conocidas como *polimorfismos de un solo nucleótido (single nucleotide polymorphism SNPs)*, lo que constituye la contribución más importante de los polimorfismos dentro de la población humana, ya que los SNPs constituyen el tipo de polimorfismo más frecuente dentro del genoma humano y, son los polimorfismos, los que indican la diferencia entre los seres humanos de cualquier origen étnico o geográfico.

En el año 2000 se terminó de secuenciar el quinto cromosoma humano completo: el cromosoma 21. Ese mismo año se terminó la secuenciación de la mosca de la fruta *Drosophila melanogaster* [8]. En México se desarrolló la secuenciación del genoma completo de la la bacteria *Rhizobium etli* [9], que es la bacteria que asociada en simbiosis con el frijol puede "fijar"el nitrógeno del aire y entregárselo a la planta. Esta secuenciación por parte de la comunidad científica mexicana se desarrolló en el Centro de Investigación sobre Fijación del Nitrógeno, en Morelos Cuernavaca, centro que ha cambiado de nombre y ahora se le conoce como *Centro de Ciencias Genómicas.*

A principios del año 2001, tanto el proyecto público del genoma humano como el proyecto privado, publicaron los primeros resultados de los trabajos realizados. Los resultados del consorcio internacional fueron publicados en la revista Nature [10] y al mismo tiempo los resultados obtenidos por los trabajos financiados por Celera Genomics se publicaron en la revista Science [11].

El 14 de abril del 2003 se anunció la culminación del Proyecto del Genoma Humano, se dió a conocer que la secuencia completa consta de 3,200 millones de pares de bases que conforman cerca de 40,000 genes [12]. Es importante señalar que la secuencia del genoma humano obtenida, es el resultado de secuenciar el ADN de diferentes individuos, por lo tanto esta secuencia es solo un estándar de la información genética contenida en cada ser humano; sin embargo, se ha demostrado que entre los seres humanos compartimos el 99.9 % de la secuencia obtenida; el 0.1 % restante varía entre cada individuo. Las variaciones más comunes son aquellas en las que cambia un sólo nucleótido (SNPs), estas variaciones se encuentran a lo largo de la cadena, en promedio una cada 800 nucleótidos y hasta el momento se han identificado cerca de 3.2 millones [12].

El conocimiento de las características genéticas de las poblaciones permite intensificar el conocimiento antropológico biológico de grupos humanos, particularmente se pueden identificar secuencias propias de estos grupos humanos que les confieren fortalezas y debilidades particulares. Estos estudios son parte de un área denominada *genética de poblaciones* y fue de las primeras en desarrollarse en México.

1.5. Genómica y salud en México

La población mexicana tiene una estructura genética particular, es el resultado de una mezcla de más de 65 grupos étnicos con grupos españoles, resultando en una estructura genética única con patrones específicos de susceptibilidad y resistencia a enfermedades comunes [12]. En general, en el campo de la salud tanto a nivel individual como a nivel de grupos humanos, el conocimiento del genoma humano ofrece nuevas formas de prevención, diagnóstico y tratamiento de distintas enfermedades, ya que se pueden detectar a aquellos individuos que tienen alto riesgo genético a desarrollar alguna enfermedad y, con esos datos, proceder a modificar su entorno con base a las características y predisposiciones individuales inscritas en el genoma.

Descripción	2000			2001			2002		
	TD**	tasa*	%	TD**	tasa*	%	TD**	tasa*	%
Total	43486	433.02	100	441004	433.09	100	457680	444.18	100
Diabetes mellitus	46,525	46.26	10.7	49,855	48.96	11.3	54,828	53.21	12.0
Enfermedades del corazón	43753	43.525	10.1	45,421	44.61	10.3	48285	46.86	10.5
Enfermedades del hígado	25378	25.3	5.8	25704	25.24	5.8	26142	25.37	5.7
Enfermedad cerebrovascular	25357	25.21	5.8	25657	25.20	5.8	26526	25.74	5.8
Tumores malignos(1)	23813	23.68	5.5	24109	23.68	5.5	24798	24.07	5.4
Causas mal definidas	8551	8.5	2.0	9195	9.03	2.1	9359	9.08	2.0
Las demás	262109	260.61	60.1	261063	256.37	59.2	26742	259.84	58.5

* tasa 1/100,000 habitantes

** Total de Defunciones (TD)

(1) Incluidos tumores malignos de tráquea, bronquios, pulmón, estómago, hígado, cuello del útero y próstata

Fuente: elaborado a partir de la base de datos de defunciones del INEGI/Secretaria de Salud[1]

Tabla 1.1: Tabla de defunciones en México en los años 2000, 2001 y 2002

En las diferentes células, los genes activos de ésta dependen de la función que ocupa la célula en un tejido en particular; sin embargo, los niveles de expresión varían de acuerdo a las condiciones del medio ambiente, estas variaciones son la respuesta a estímulos ajenos al genoma como la temperatura, el alimento y los rayos solares; de la misma manera, los organismos responden a través de la expresión de su genoma ante el contacto con fármacos, infecciones o diferentes estados patológicos. Es por ello, que con herramientas que permitan comparar los niveles de expresión de células en diferentes estados, se puede conocer la respuesta genómica a las enfermedades de un organismo, lo que puede derivar en un diagnóstico genéticamente preciso y nuevas estrategias en el diseño de tratamientos que actúen de una manera específica.

En México, según datos de la Secretaría de Salud[2], la principal causa de muerte en el país es la diabetes mellitus (Tabla 1.1), que es la incapacidad o deficiencia de producir insulina o la dificultad para que la insulina que logra producir una persona actúe dentro de las células, lo que provoca que los niveles de glucosa en el organismo se eleven, ya que es la insulina la que se encarga de controlar los niveles de glucosa en la sangre.

Enfermedades como la diabetes no sólo generan un gran número de pérdidas humanas, aunque es el factor más importante, estas enfermedades también generan gastos económicos enormes, ya que el costo de los tratamientos suelen ser elevados. Esta es una de las reazones por las que se decidió incursionar en el área de la medicina genómica en el país. Debido a que los costos de prevención de las enfermedades más frecuentes comparada con los costos de tratamiento crónico y rehabilitación son menores, se llegó a la conclusión de que el impacto de la incursión de la medicina genómica en México no solo se reflejaría en un mejor estado de salud de la población, también traerá consigo un impacto económico favorable para el país. [7]

[2] http://salud.gob.mx

Muchas investigaciopnes elaboradas en torno al tema de la diabetes se han centrado en el comportamiento de los genes que están presentes en las células de organismos diabéticos. Investigadores belgas identificaron en el 2001 un gen implicado en la regulación de la insulina [13], las especulaciones de los investigadores señalan que la alteración de dicho gen puede hacer a las personas más sensibles a los efectos de la insulina, evitando muchas dosis de esta sustancia. De la misma manera, investigadores de la Universidad de Chicago, encontraron un gen encargado de la fabricación de una enzima que regula los niveles de glucosa en la sangre y encontraron una asociación entre la susceptibilidad de la diabetes y dicho gen entre la población mexicana [14].

Al igual que con la diabetes, se han realizado diversos estudios a nivel genético sobre diferentes enfermedades, una de ellas el cáncer. El conocimiento de las características genéticas de algunos grupos humanos permite el conocimiento de las inmunidades así como las susceptibilidades que, a nivel de grupo, hacen característica a una población. En México, el conocimiento de algunas características genéticas de los grupos étnicos de nuestro país tendrá un gran impacto en el conocimiento antropológico de los mexicanos y nos va a permitir también conocer la historia desde un punto de vista molecular.

En México, la genética humana ha tenido un desarrollo reciente. A finales de los años cuarenta aparecieron las primeras publicaciones sobre la distribución de los grupos sanguíneos en poblaciones mexicanas.

En los años sesenta se formaron los primeros grupos de genética en hospitales de la Ciudad de México y en Guadalajara, poco después surgió la Asociación Mexicana de Genética Humana, así como la Especialidad en Genética Humana en el Centro Médico Nacional del Instituto Mexicano del Seguro Social y el Consejo Nacional de Especialistas en Genética Humana, que en la actualidad se denomina Consejo Mexicano de Genética.

Aunque México no participó de manera directa en el esclarecimiento del genoma humano, en nuestro país no se deben dejar de aprovechar los beneficios de la genómica. Mediante su aplicación se fortalecerá el desarrollo de la investigación científica y el desarrollo tecnológico e industrial de las empresas mexicanas en el campo farmacéutico biotecnológico y de la prestación moderna de los servicios de salud, así como también habrá de incidir en la disminución de los costos de atención médica [15].

1.5.1. INMEGEN

La Fundación Mexicana para la Salud (FUNSALUD) organizó a principios de 1999 un grupo de trabajo, integrado por especialistas de la UNAM y de los Institutos Nacionales de salud, que se dedicó a analizar qué posibilidades se tendrían para aprovechar la información y el conocimiento que se derivaría del esclarecimiento del genoma humano, siempre a favor de la salud de los mexicanos [15]. En octubre del año 2000 se formalizó un convenio entre la Secretaría de Salud, la UNAM, FUNSALUD y el

CONACyT, documento en el cual se estableció el compromiso de crear un centro de medicina genómica. El primer paso, descrito mediante el convenio, era el estudio de la factibilidad técnica, económica, social y política para el desarrollo de dicho centro, posteriormente se estableció un grupo formado por representantes de las instituciones firmantes en el convenio, el cual se responsabilizó de supervisar la elaboración del estudio mencionado.

El *Estudio de la factibilidad para el establecimiento y desarrollo del Centro de Medicina Genómica* se entregó en agosto del 2001. Los resultados expuestos sobre el estudio realizado recibieron la aprobación respectiva y se decidió continuar con la segunda etapa: la creación de un instituto de medicina genómica en México.

En noviembre del 2001, los titulares de las cuatro instituciones participantes firmaron un nuevo convenio para establecer el *Consorcio Promotor del INMEGEN*, el cual tiene como objetivo principal coordinar las acciones académicas, financieras y legales referentes a la organización para formar el INMEGEN y promover su vinculación con Instituciones Nacionales e Internacionales que puedan ayudar a llevar a cabo los programas de investigación, formación de recursos humanos y difusión del conocimiento de la medicina genómica, a través de la articulación de proyectos. Desde esa ocasión se decidió que la futura organización se ubique dentro del sector salud y llegue a tener el perfil de un Instituto Nacional de Salud [15]. Desde enero del 2002 funciona este Consorcio Promotor del INMEGEN.

El 19 de julio se dió a conocer oficialmente la creación del INMEGEN, este Instituto tienen como misión contribuir a generar y aplicar el conocimiento derivado del esclarecimiento del genoma humano. La asistencia médica se desarrollará a través de otras Instituciones de Salud.

Con toda la información obtenida del genoma humano, dado que ésta se encuentra íntimamente relacionada con la información de cada individuo, no es difícil imaginar que las implicaciones éticas, sociales y políticas se incrementarán conforme pase el tiempo y las investigaciones sigan avanzando. Esto es, porque en el ámbito social, en donde son considerados los derechos humanos, el debate no se ha desarrollado de la misma manera que los avances científicos y tecnológicos en torno a la información obtenida del genoma humano.

Capítulo 2

Microarreglos de ADN

Cuando se estudia un gen, la metodología habitual es estudiar cómo se manifiesta éste en el modelo biológico estudiado (qué proteína codifica), determinar la relación que tiene con otros genes y diseñar modificadores de la expresión (inhibidores o activadores) para regular su efecto en condiciones patológicas.

La metodología descrita anteriormente no resulta muy eficiente si consideramos la cantidad de genes que puede existir en un organismo complejo (como el ser humano), pues pueden aumentar de manera considerable las relaciones gen-gen, gen-proteína y proteína-proteína, retrasando los resultados de dichas investigaciones. Este es el motivo principal para que se busque un método masivo para el análisis de genes. Los *microarreglos de ADN* permiten realizar dicho análisis masivo, reduciendo tiempo y costo de cada experimento.

Un *microarreglo de ADN* puede ser visto como un conjunto ordenado de genes sobre una pequeña superficie [16] a la que se le conoce como *chip*. Esta técnica fué desarrollada en 1999 por Patrick O. Brown y David Botstein [16] y es posible trabajar con todos los genes de un organismo determinado en un solo experimento.

2.1. Impresión de microarreglos

Para elaborar un microarreglo primero se deben imprimir los genes sobre una superficie, los genes que se van a imprimir pertenecen a diferentes *bibliotecas genómicas* y éstas se escogen de acuerdo al organismo del cual se va a hacer el experimento, es decir, en el mismo arreglo los genes impresos deben pertenecer a la misma biblioteca.

Una biblioteca genómica es una colección de fragmentos de ADN genómico (ADNg) clonados en un vector, que en conjunto representan el genoma de un organismo, estos

fragmentos de secuencias se denominan *sondas*. En una biblioteca genómica es posible seleccionar fragmentos de ADN entre millones de secuencias clonadas en cantidad suficiente para ser analizada. Los fragmentos clonados deben ser de tamaño suficiente para contener, en lo posible, genes con sus secuencias reguladoras, pero no tan grandes que dificulten su discriminación.

Para tener una biblioteca genómica se debe contar con una técnica de clonación eficiente para poder construir las bibliotecas. Uno de los métodos que se usa comúnmente es la *Reacción en Cadena de la Polimerasa*, PCR por sus siglas en inglés, además se debe tener un sistema de selección de clones para obtener secuencias específicas dentro de toda la colección.

La PCR es un método *in vitro* de síntesis de ADN con el que un segmento en particular de éste es específicamente amplificado al ser delimitado por un par de cebadores o iniciadores que lo flanquean. Su copiado se logra de manera exponencial a través de repetidos ciclos de diferentes periodos y temperaturas de incubación en presencia de una enzima ADN polimerasa termoestable. Así se obtienen en cuestión de horas millones de copias de la secuencia deseada de ADN [17].

En una biblioteca genómica, los clones son colocados en posillos ordenados en arreglos para poder identificar a las secuencias clonadas, para elaborar los microarreglos de ADN se requiere contar con las bibliotecas adecuadas. La identificación de las diferentes secuencias en los posillos se da por la ubicación de cada posillo, este ordenamiento se verá reflejado en el microarreglo generado por el impresor. Hay que mencionar que algunos de los pozos dentro de las bibliotecas pueden no contener clones de genes, a estos pozos se les conoce como *pozos vacíos*. El porcentaje de los pozos vacíos se debe divulgar como una característica de a biblioteca [18].

2.1.1. Impresores

Para elaborar un microarreglo, primero se deben imprimir las sondas sobre una superficie. Para hacer la impresión se requiere de un aparato conocido como *impresor*, el cual toma una microscópica cantidad de cada pozo de las bibliotecas y lo traslada a una pequeña superficie de manera ordenada, generando poco a poco el arreglo de genes con el que se va a hacer el experimento.

En la actualidad existen varios tipos de impresores de microarreglos, los más sencillos son los de contacto (pin printing o microspotting), que como su nombre lo indica, depositan las muestras haciendo contacto con la superficie. Las agujas o pins se empapan en la solución que contiene la muestra, con lo que una pequeña cantidad de ésta se transfiere al extremo de la aguja. Una gota de la muestra queda impresa cuando los extremos de las agujas tocan la superficie del chip. Este tipo de impresores son los más económicos y su uso es relativamente sencillo. Básicamente se trata de un brazo robótico con movimientos en los ejes X, Y y Z y su característica más importante es poderse desplazar en cualquiera de estas direcciones con una resolución de una décima

de milímetro (Figura 2.1) [16].

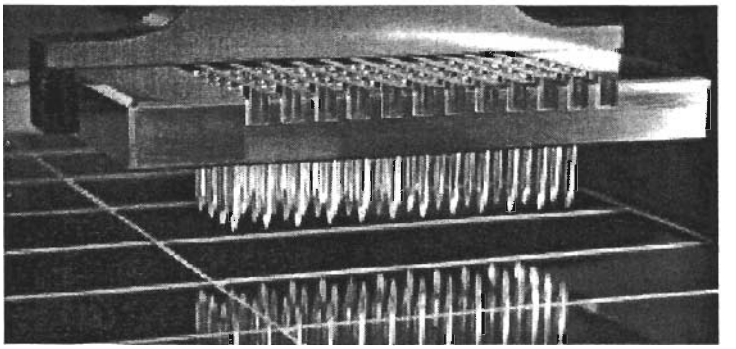

Figura 2.1: Impresor de contacto. Tomado de [16]

Las agujas del impresor van a formar pequeños arreglos sobre la laminilla al momento de imprimir las muestras. El conjunto de estos pequeños arreglos forman el microarreglo completo y se conocen como *(subgrids)*. Para identificar un gen en el microarreglo se debe saber su posición dentro del mismo, es decir, las coordenadas renglón/columna en el microarreglo (*metarenglón*, *metacolumna*) y las coordenadas renglón/columna en el subgrid (*renglón*, *columna*).

2.1.2. Fijación de las sondas

Una vez que son impresas las laminillas, las sondas deben ser fijadas o inmovilizadas sobre la superficie del microarreglo, para esto se han desarrollado múltiples procedimientos que dependen de la aplicación final del microarreglo y, en gran medida, del tipo de soporte utilizado.

En uno de los métodos de fijación de sondas, las laminillas impresas se hornean a una temperatura aproximada de 80°C por cuatro horas. Adicionalmente se delimita la región en donde se encuentra el microareglo y se identifica cada laminilla, ya sea con un código de barras o con un número de serie [16]. En este paso, las sondas se desnaturalizan y sólo se fija a la laminilla una de las hebras que conforman el segmento de secuencia, esto es muy importante para que el desarrollo del experimento pueda llevarse a cabo correctamente.

2.1.3. Marcaje de sondas

Un microarreglo de ADN nos permite identificar cómo se comportan los niveles de expresión de un organismo bajo cierta condición *(Experimental)* con respecto a otra

(Control). El principio básico de la tecnología de los microarreglos se basa en la propiedad que tienen los nucleótidos de hibridar específicamente con otros [19], de hecho, las sondas pertenecientes a cada una de las muestras (control y experimental) deben ser marcadas para posteriormente hibridar con las sondas fijadas en el microarreglo.

Existen diferentes métodos para realizar este marcaje, pero los más utilizados son los químicos o enzimáticos, utilizándose comúnmente los fluoróforos Cy3 y Cy5 (cianinas). El uso de fluoróforos permite analizar varias muestras en el mismo arreglo, eliminando la variabilidad que se produce cuando se comparan los resultados de la hibridación de diferentes arreglos, además de hacer posible el escaneado de múltiples marcadores simultáneamente [18]; sin embargo presentan dos inconvenientes:

- En un microarreglo sólo se pueden analizar dos patrones de expresión de genes, y además el patrón de incorporación de los fluoróforos no es uniforme
- El gran tamaño de los fluoróforos contribuye a dificultar la incorporación enzimática de los nucleótidos que compondrán la sonda.

Antes de hacer el marcaje de las sondas es necesario convertir el ARNm de las muestras en *ADNc* (ADN complementario) para que posteriormente puedan hibridar con las sondas fijadas en el microarreglo. El proceso mediante el cual se hace esta transformación se llama *transcripción inversa*. El ADNc es marcado con un fluoróforo diferente para cada una de las muestras, si las sondas control son marcadas con el fluoróforo Cy3, entonces las sondas control deben ser marcadas con el fluoróforo Cy5.

2.1.4. Hibridación

Una vez marcadas las sondas, se procede a realizar la hibridación de éstas con las sondas fijadas en la laminilla del experimento.

Se vierten las sondas marcadas sobre la laminilla y se espera que hibriden, posteriormente se hace el lavado para eliminar los excesos, es decir, remover las sondas marcadas que no hayan hibridado con algún blanco; sin embargo hay que tener cuidado en este paso del experimento para no eliminar sondas que sí hayan hibridado.

2.2. Lectura de los microarreglos

Una vez generado el microarreglo, es necesario someterlo a un proceso de análisis que nos permita obtener de manera clara que sondas aumentaron, disminuyeron o no presentaron cambios en sus niveles de expresión.

Los fluoróforos que se utilizaron para marcar las sondas, emiten luz al ser exitados

mediante una determinada longitud de onda (láser). [18]. Los brillos emitidos por cada uno de los fluoróforos son medidos por *lectores de fluorescencia*.

Existen dos tipos de lectores de microarreglos: los lectores con cámaras digitales y los lectores confocales, pero ambos tipos de lectores utilizan un láser para excitar las moléculas fluorescentes unidas a las sondas blanco y poder obtener la imagen de cada una de las muestras contenidas en el microarreglo. Este procedimiento se hace para cada uno de los fluoróforos obteniéndose dos imágenes: una para el fluoróforo Cy3 (rojo) y otra para el fluoróforo Cy5 (verde) [16], con estas imágenes se pueden observar a grandes rasgos los cambios en los niveles de expresión.

Para la obtención de cada una de las imágenes se debe ajustar la intensidad del láser y la sensibilidad de la cámara, de tal manera que ambas imágenes den valores semejantes de fluorescencia total. Comúnmente para lograr esto, en los microarreglos se colocan controles o marcadores que se utilizan para ajustar la señal de la fluorescencia de ambos fluoróforos [16]. Una vez obtenidas estas imágenes, pueden ser combinadas para obtener un aspecto visual del microarreglo (Figura 2.2).

En la imagen combinada se pueden observar puntos verdes, rojos y amarillos y un sinnúmero de tonalidades entre el verde y el rojo. Si suponemos que la muestra experimental ha sido marcada de rojo y la control de verde, todos aquellos puntos que se ven rojos o en tonalidades anaranjadas, pueden ser interpretados como genes que aumentaron su expresión en la muestra experimental. Mientras que los puntos verdes o en tonalidades entre el amarillo y el verde serán aquellos genes que disminuyeron su expresión. Finalmente los puntos amarillos representan a los genes que en ambas condiciones se expresan de igual forma.

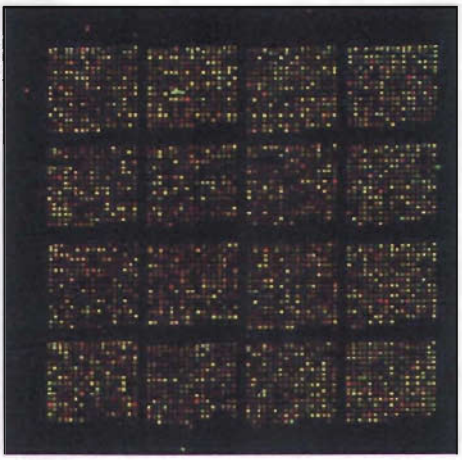

Figura 2.2: Imagen de fluorescencia combinada de un microarreglo. Tomado de [16]

2.2.1. Cuantificación de microarreglos

Para la obtención de los valores cuantitativos de cada una de las señales de fluorescencia contenidas en el microarreglo, se requiere del análisis de las imágenes que se obtienen con el láser.

En primer lugar se debe considerar la aplicación de un filtro para depurar la imagen de pequeñas imperfecciones o señales no deseadas que pudieran encontrarse entre las muestras. Posteriormente se genera una retícula en la que se definen las áreas que se van a cuantificar (Figura 2.3). Definida la retícula, se determina una zona de exclusión y el área para calcular el fondo de la imagen (*background*) (Ver recuadro de la Figura 2.3). Definidos estos parámetros se calcula la densidad de los pixeles en cada área definida, dando como resultado una tabla con las coordenadas, los valores de la densidad (fondo y señal) para cada una de las muestras en el microarreglo (Tabla 2.1). Esta información es complementada con las bases de datos en donde se tiene el nombre del gen, su función, la vía metabólica a la que pertenece, ubicación celular, etc.

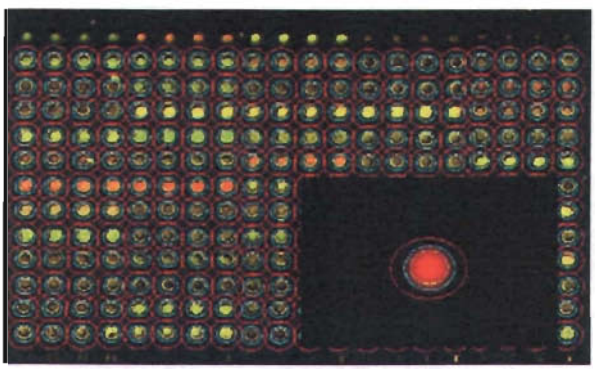

Figura 2.3: Marcaje del ruido de Fondo. Delimitación de la zona a considerar como ruido de fondo. Tomado de [16]

ORF-Ref	Name	biological process	molecular function	Density [A]	Density [B]	Bckgd [A]	Bckgd [B]	Signal [A]	Signal [B]
YCL050C	APA1	biological process	ATP adenylyltransfer	8468	7662	2782	4134.8	5686.1	3526.74
YCL050C	APA1	biological process	ATP adenylyltransfer	9169	7187	2872	4176.9	6296.84	3010.35
YCR046C	IMG1	protein biosynthesis	structural constituent	5766	5344	2923	4160.3	2843.06	1183.56
YCR046C	IMG1	protein biosynthesis	structural constituent	5468	5728	2975	4233.8	2493.1	1494.17
YCR023C		biological process	molecular function	4064	4452	3096	4335.9	968.201	115.816
YCR023C		biological process	molecular function	3922	4402	2877	4155.6	1044.91	246.366
YIL130W	GIN1	biological process	molecular function	10678	12578	3172	4492.6	7506.54	8085.82
YIL130W	GIN1	biological process	molecular function	9317	10665	3127	4423.6	6189.82	6241.04
YGL197W	MDS3	sporulation	molecular function	13727	18619	2982	4344.5	10744.4	14274.6
YGL197W	MDS3	sporulation	molecular function	12506	16782	2894	3957.1	9611.54	12825.4
YIL154W	IMP2	DNA repair*	transcription factor	14830	13706	2855	4147.4	11975.3	9558.31
YIL154W	IMP2	DNA repair*	transcription factor	15377	14062	3112	4648	12265.7	9413.55

Tabla 2.1: Fragmento del archivo que contiene la información cuantificada después del proceso de análisis de la imágen del microarreglo.

Capítulo 3

Análisis de microarreglos

Independientemente del diseño y la tecnología que se haya utilizado para elaborar un microarreglo, el objetivo es explorar las relaciones entre los genes, así como encontrar patrones en los niveles de expresión. Antes de que estos niveles de expresión puedan ser comparados de manera correcta, se deben llevar a cabo ciertas rutinas de *normalización* y *transformación* sobre los datos para eliminar medidas cuestionables y de baja calidad ocasionadas por diversas fuentes de variación que son inherentes al proceso experimental. De esta manera, se espera ajustar las intensidades para posteriores comparaciones y finalmente para seleccionar los genes que se encuentran significativa y diferencialmente expresados.

3.1. Razón de expresión

Cada experimento consta de N elementos denominados *spots*, en donde se comparan 2 muestras: una experimental y una control y cada una de las muestras va a estar representada por un fluoróforo dependiendo del experimento. Para obtener una primera relación entre ambas muestras se obtiene el *valor de la razón (ratio)*. Suponiendo que la muestra experimental está representada por el fluoróforo Cy3 y la muestra control con el fluoróforo Cy5, entonces la razón de la expresion para el gen i=1...N se obtiene como se muestra en la ecuación 3.1, en donde Cy3 representa la intensidad bajo la condición control y Cy5 la intensidad bajo la condición experimental:

$$Razon_i = \frac{Cy5_i}{Cy3_i} \tag{3.1}$$

Al calcular la razón de la expresión se está midiendo el cambio de expresión que existió en el experimento; sin embargo, no se pueden distinguir aquellos genes que

aumentaron o disminuyeron su expresión con respecto a la muestra control (Cy3). Para resolver este problema, se calcula el logaritmo en base 2 de la razón para cada spot en el microarreglo, este valor se denomina *R*.

$$R_i = log_2(\frac{Cy5_i}{Cy3_i}) \tag{3.2}$$

Se dice que un gen está sobre-expresado si su nivel de expresión bajo la condición experimental es mayor que su nivel de expresión bajo la condición control, es decir $Cy5 > Cy3$, en este caso su valor R será mayor que cero. Un gen entonces está sub-expresado si su nivel de expresión bajo la condición experimental es menor que su nivel de expresión bajo la condición control ($Cy5 < Cy3$). El valor R es cero cuando no existe un cambio de expresión, es decir, cuando $Cy3 = Cy3$. Obtener el valor R para cada spot en el microarreglo nos dá un punto de referencia para identificar a los genes sobre-expresados y sub-expresados.

Al obtener el logaritmo de la razón de la expresión se obtiene una simetría en la gráfica de frecuencias con respecto al cero en comparación con la gráfica de frecuencias de la razón (Figura 3.1).

3.2. Corrección del Background

La señal reportada en los archivos de datos que provienen de la lectura de un microarreglo, en realidad corresponde a la suma del background y de la expresión genética real. Mediante el proceso de *corrección del background* se pretende acercar los datos registrados lo más posible al valor real de la señal, con el fin de que los datos con los que se está trabajando tengan un mayor significado biológico.

Existen muchos procedimientos para llevar a cabo la corrección del background; sin embargo uno de los más sencillos se basa en hacer la sustracción para cada una de las señales del valor de su background registrado en los archivos por cada spot del arreglo, con la siguiente fórmula:

$$Cy3_i = Cy3_i - BackgroundCy3_i$$

$$Cy5_i = Cy5_i - BackgroundCy5_i \tag{3.3}$$

Esta forma de proceder es la más sencilla, sin embargo se ignoran ciertos errores que pueden surgir en la fase experimental que pueden llevar a obtener valores falsos tras hacer la corrección del background, tales como:

- La lectura del background puede ser mayor que la señal del spot, llevando a valores negativos en la expresión real.

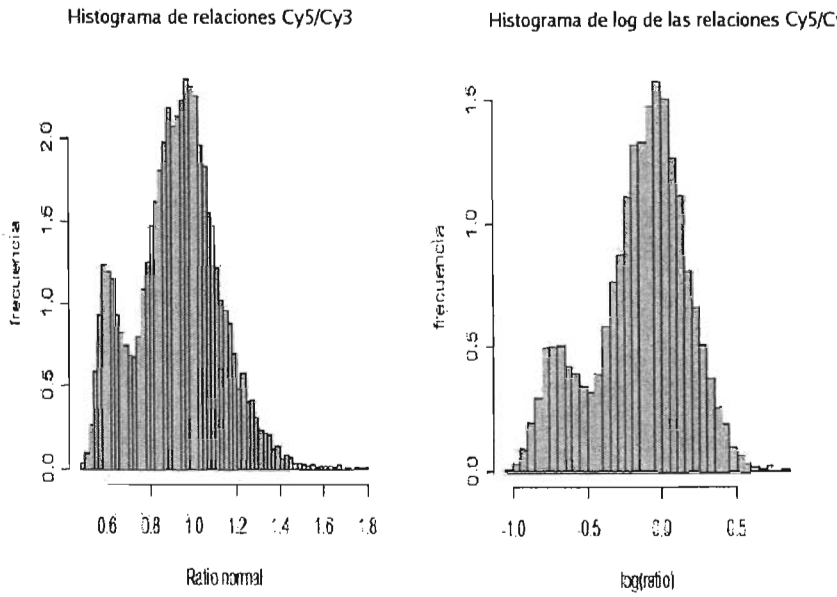

Figura 3.1: Comparación de gráfica de frecuencias de la relación Cy5/Cy3 de un experimento de microarreglos: ratio.

- Puede haber errores al definir el valor del background para cada spot. Si el valor del background es muy alto, al hacer la corrección bajo este criterio cambiaría la expresión real de la señal del spot quitándole significado biológico.

Al hacer la sustracción de valores para corregir el background se pueden obtener valores que no corresponden a la intensidad real de las señales restando significado biológico al experimento, es por ello que algunos investigadores optan por no aplicar esta transformación.

3.3. Normalización

Independientemente de haber hecho o no la corrección del background, la primera transformación que se aplica a los datos del arreglo para que puedan compararse unos con otros es conocida como *normalización* [18]. El objetivo de hacer una nor-

malización es identificar, eliminar y corregir fuentes de variación sistemática que no sean variaciones biológicas para ajustar ambas señales. Estas variaciones pueden ser el resultado de algunos errores presentados durante la fase experimental como:

- Presencia de polvo durante la hibridación
- Falta de homogeneidad en la superficie del soporte o slide
- Cantidad desigual de sondas
- Diferencias en la incorporación de fluoróforos
- Errores en el escaneo de la imagen

Para poder ajustar las intensidades entre Cy3 y Cy5, en general se necesita incrementar un poco la intensidad de Cy5 con respecto a Cy3, debido a que la estructura molecular de Cy5 es más grande que la de Cy3, lo que causa que se incorpore con menor eficiencia. Para esto, se debe multiplicar Cy5 por el factor de ajuste obtenido con el algoritmo de normalización.

Aunque este proceso es únicamente un paso intermedio en el análisis, tiene una influencia considerable sobre los resultados finales, ya que de no hacerlo, los errores sistemáticos mencionados anteriormente se mantienen en gran medida.

3.3.1. Normalización con lowess

Muchos de los algoritmos de normalización que son aplicados de manera global a los datos asumen que el total de fluorescencia en ambas señales es igual. Bajo esta suposición, obtienen una constante de normalización que es usada para ajustar las intensidades de fluorescencia de ambas señales.

La normalización de manera global resulta muy sensible a un pequeño porcentaje de genes diferencialmente expresados que tienen valores extremos de las intensidades. Un método para calcular el factor de normalización que puede remover esta dependencia que tienen las intensidades sobre los valores del $log_2(ratio)$ y que ha demostrado ser bastante eficiente se conoce como LOWESS (Locally weighted linear regression). Para visualizar los efectos de esta dependencia de las intensidades y describir el resultado de aplicar lowess se necesitan dos valores:

$$R_i = log_2(Cy5_i/Cy3_i)$$

$$I_i = log_{10}(Cy5_i * Cy3_i) \tag{3.4}$$

El valor R corresponde al ratio y el I al producto de las intensidades. Al graficar el valor R contra I de todos los elementos del arreglo, se pueden observar los efectos específicos de la intensidad en los valores de R (Figura 3.2 (A)).

El algoritmo de LOWESS consiste en crear una ventana deslizante que se va a recorriendo de izquierda a derecha. Mientras avanza esta ventana se calcula un polinomio de orden bajo (en el caso de genArise de orden 2) para cada elemento contenido en la ventana. A cada punto se le asigna un peso dependiendo de la distancia a la que se encuentre del punto de regresión (x). Disminuyendo así la sensibilidad a los valores extremos.

El algoritmo de normalización con LOWESS considera que la corrección que debe aplicarse a los datos es función de la intensidad de las señales analizadas (Cy3 ó Cy5). El método consiste en ajustar los datos de Cy3 ó Cy5 de cada spot por los factores calculados con el ajuste. LOWESS usa una función de pesos que hace énfasis en las contribuciones de los datos de los elementos del arreglo que están lejos de cada punto en la gráfica *R-I*. Los resultados de aplicar una corrección de LOWESS se pueden ver en la Figura 3.2 (B).

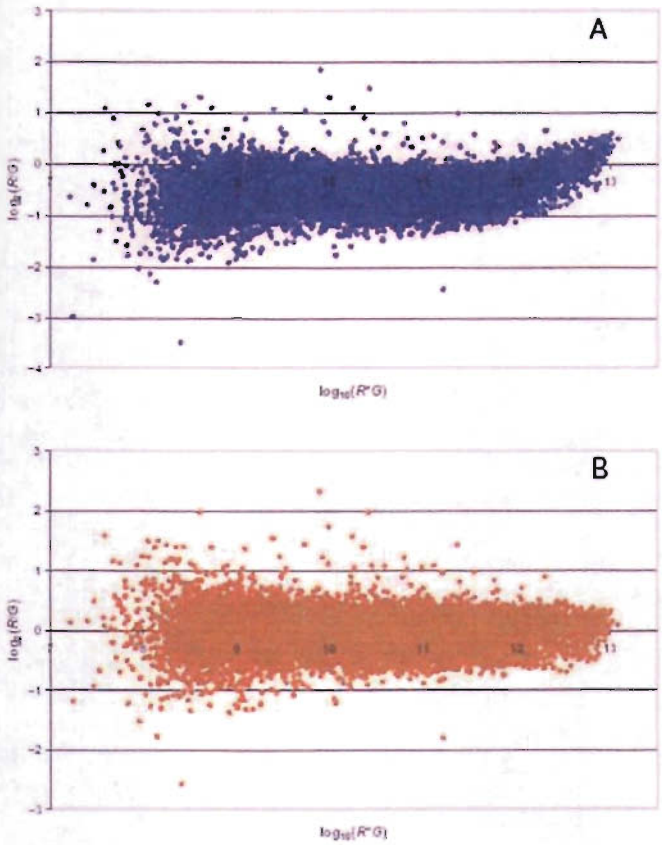

Figura 3.2: Gráfica R-I antes de aplicarles cualquier transformación. Tomado de [20]

La normalización por lowess aplicada al conjunto total de datos se conoce como *normalización global*; sin embargo, se asume que existen variaciones sistemáticas que pueden ser originadas por la separación de las agujas del aplicador del impresor. Con el fin de hacer una corrección más precisa se genera una curva de regresión o ajuste para cada subarreglo (subgrid) del microarreglo, asumiendo que la dispersión de los datos entre las agujas para el subgrid debe ser la misma. A esta forma de aplicar la normalización se le conoce como *normalización local*.

El factor de normalización obtenido por LOWESS finalmente es utilizado para obtener los nuevos valores de las intensidades que representan los valores normalizados, mediante la siguiente ecuación:

$$Cy3' = Cy3 \text{ y}$$

$$Cy5' = Cy5 * 2^{y(x_i)} \tag{3.5}$$

en donde $y(x_i)$ es el factor de ajuste obtenido por lowess

Esta acuación se deriva de que:

$$log_2(T_i') = log_2(T_i) - y(x_i) \Rightarrow log_2(T_i) - log_2(2^{y(x_i)}) \tag{3.6}$$

3.4. Filtrado

Al analizar la gráfica R vs I, se puede detectar que la variabilidad de los valores de R aumenta conforme disminuye el valor de la suma de las intensidades (I). Esto se debe a que el error relativo aumenta conforme disminuye el valor de las intensidades, en este caso, el valor de la señal se acerca al valor de su respectivo background. Para corregir este problema, la opción es usar únicamente los elementos del arreglo que tienen un valor de señal significativamente diferente al valor del background respectivo, es decir, eliminar aquellos elementos del arreglo en donde no es posible distinguir entre las intensidades y el posible ruido del background correspondiente.

Si se mide la media del background local cercano a cada elemento del arreglo y su desviación standard, se espera un 95.5 % de confianza de que los elementos a considerar tendrán intensidades mayores a dos desviaciones standard arriba del background [20], es por ello que se conservan únicamente los elementos del arreglo que satisfacen la siguiente condición, en donde σ representa la desviación estándar, de los backgrounds correspondientes:

$$Cy5_i > 2 * \sigma(Background(Cy5_i)) \text{ ó}$$

$$Cy3_i > 2 * \sigma(Background(Cy3_i)) \tag{3.7}$$

Otros enfoques incluyen el uso de un *límite (threshold)* para los elementos aceptables del arreglo o un *cut-off* basado en algún porcentaje en el cual una fracción de los elementos es descartada [20]; sin embargo, al establecer un límite de manera arbitraria pueden ocurrir dos problemas:

- Si se establece un threshold muy pequeño, puede suceder que no se eliminen elementos con poca o nula significancia biológica para el análisis
- Si se establece un threshold muy grande, se pueden eliminar elementos que resulten de interés para el análisis, incurriendo así en pérdida de información

Al definir el límite a partir del valor del background de cada una de las señales, disminuye el riesgo de perder elementos con un mayor significado biológico.

3.5. Análisis de duplicados

La duplicación de las sondas dentro de un microarreglo es una medida esencial para medir la confiabilidad del experimento. Se espera que los valores, tanto de las intensidades y del background, de los duplicados de la misma sonda no varíen de manera significativa pues puede proporcionar una medida variabilidad biológica natural del sistema estudiado, en un experimento ideal se espera que los valores de la sonda duplicada sean iguales.

Con la finalidad de mantener un valor representativo por cada gen, se debe llevar a cabo un análisis de duplicados con el objetivo de aplicar una transformación, bajo un cierto criterio, a las observaciones duplicadas y conservar un valor por cada gen. En esta fase se eliminan elementos que resulten cuestionables o de baja calidad, ya que al tratarse del mismo gen se espera que el valor de la relación Cy5/Cy3 no varíe demasiado, de modo que, si los duplicados presentan una coloración diferente en la imagen combinada de fluorescencias, esos serán considerados elementos cuestionables (Figura 3.3).

Para este propósito también existen diferentes métodos, los más utilizados es la obtención de la media y la media geométrica para los elementos repetidos.

Aunque en ambos casos se obtiene un valor representativo para cada conjunto de genes repetidos, al utilizar la media geométrica se obtienen valores más confiables que al utilizar únicamente la media o promedio, esto sucede, porque la media geométrica es menos sensible a valores extremos que el promedio, debido su fórmula denotada por:

$$mediaGeometrica = \sqrt[N]{\Pi_1^N a_i} \tag{3.8}$$

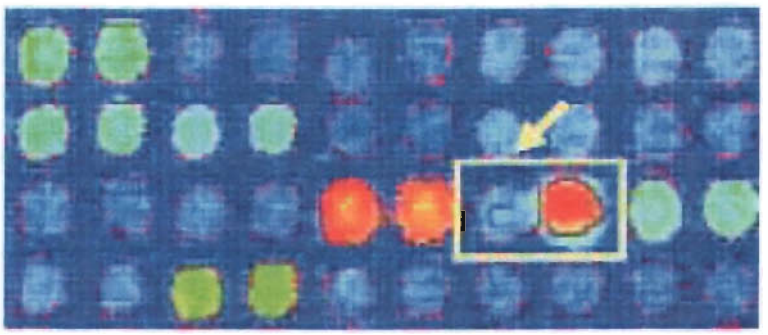

Figura 3.3: Elementos cuestionables.

En donde N representa el número de elementos repetidos, Π el producto y a_i el valor a del spot i, es decir la media geométrica es el resultado de obtener la raíz *N-ésima* del producto de N valores, en este caso se obtiene el valor de la media geométrica para cada una de las intensidades del gen duplicado. Tras haber realizado el análisis de duplicados, si en el microarreglo se contaba con sondas duplicadas, el número de elementos debe ser la mitad de los originales si es que en alguna otra etapa del preprocesamiento no se eliminaron elementos de baja calidad para el análisis.

3.6. Identificación de los genes diferencialmente expresados

Una vez elimiadas las posibles fuentes de variación sistemática, es necesario identificar a los genes que se encuentran diferencialmente expresados de manera significativa, para esto es necesario obtener una medida estadística de estandarización que permita ponderar los valores y permitir un criterio discriminante para obtener aquellos genes que se encuentran sobre-expresados y sub-expresados.

La medida de estandarización que se utiliza es *z-score*, el cual nos indica a cuántas desviaciones estándar está cada valor dentro de nuestro conjunto de datos con respecto a la media, y está denotado con la siguiente fórmula:

$$z_i = \frac{x_i - mean(x)}{\sigma(x)} \tag{3.9}$$

Dada la gráfica 3.4 que muestra la distribución normal estándar de un conjunto de datos, por propiedades de la desviación estándar (σ), se obtiene un 95 % de certeza de que los genes diferencialmente expresados de manera más significativa serán aquellos

que tienen un valor:

$$|z - score_i| > 2 \tag{3.10}$$

Con i=1...N, donde N es el número total de datos.

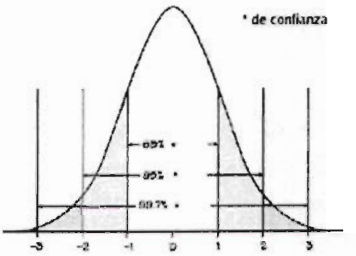

Figura 3.4: Gráfica de una distribución normal estándar en donde se muestran los porcentajes de confiabilidad que tiene los puntos tras haber obtenido su valor del z-score. Tomado de www.ltcconline.net

Si el valor de la media y de la desviación estándar se toma con respecto al conjunto total de genes, tras un análisis de la gráfica RvsI se podría suponer que este enfoque no se refleja directamente en la estructura de los datos, por lo siguiente:

- Cuando las intensidades tienen valores muy bajos y los datos son mucho más variables, se pueden perder genes que en realidad son diferencialmente expresados
- Por el contrario, cuando las intensidades tienen valores muy altos, los genes que son significativamente expresados podrían no ser identificados

Un enfoque alternativo consiste en usar la estructura local del conjunto de datos, dicho algoritmo es conocido como *ventana deslizante*(sliding window) [20], en donde el valor de z-score es calculado para cada subconjunto de datos y no para el conjunto total, que se determina por una ventana de cierto número de elementos con características similares[1]. El tamaño de la ventana (número de elementos) no debe ser muy grande pues se puede incurrir en la pérdida de significado biológico como se mencionó anteriormente; sin embargo no puede ser muy pequeño pues no se podrían identificar los genes expresados de manera significativa.

El tamaño de las ventanas que definimos por defecto para el sistema es de 50 elementos [21], sin embargo es posible definir un tamaño de ventana elegido por el usuario.

[1] En la fórmula 3.9, x debe representar los elementos de la ventana

Los valores deben ser ordenados con respecto a su intensidad en ambos canales I ($log_{10}(Cy5_i * Cy3_i)$). Si el tamaño de la ventana está definido por N, una vez ordenados los elementos se toman conjuntos de N elementos para cada spot en el microarreglo (x_i) bajo la siguiente restricción:

- Los primeros N/2 elementos, se encuentran en una ventana que abarca a los primeros N datos.
- Los últimos N/2 elementos, se encuentran en una ventana que abarca a los últimos N datos.
- Los otros spots comparten en su ventana a los N/2 menores y a los N/2 elementos con un valor ratio mayor

Al obtener el valor del Z-score, obtenemos en realidad una referencia para determinar qué elementos están sobre-expresados y qué elementos están sub-expresados bajo un cierto número de desviaciones estándar. Generalmente, los elementos que obtienen un valor z-score positivo estarán sobre-expresados una vez que se haya determinado el intervalo de desviaciones estándar como punto de referencia.

Si graficamos los valores R-I, pero coloreamos los puntos de la grafica de acuerdo al valor del z-score obtenemos una grafica como la que se muestra en la figura 3.5.

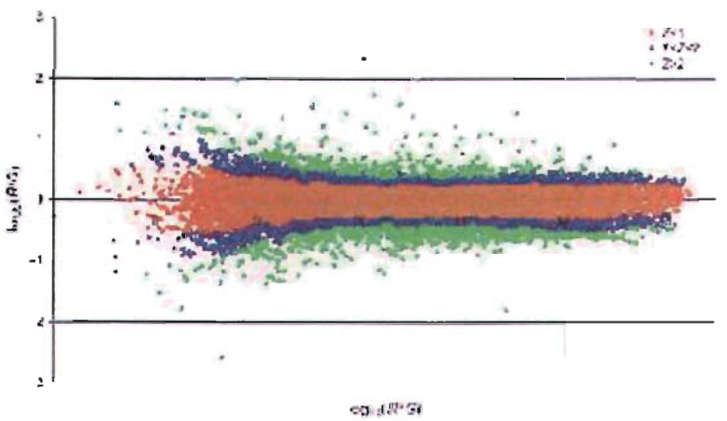

Figura 3.5: Gráfica de los valores del Z-score. Tomado de [20]

Una vez obtenidos los conjuntos de los genes que se encuentran sobre-expresados y sub-expresados, se debe determinar el significado biológico de esta información.

Capítulo 4

genArise

4.1. Motivación

El análisis de microarreglos al ser una tecnología relativamente nueva, el desarrollo de las herramientas que faciliten este análisis y se adapten a las necesidades del investigador han tenido un avance todavía más reciente; sin embargo, muchas de ellas tienen un costo muy elevado, por ejemplo: *GeneSpring version 7.2* cuya licencia se debe pagar de manera anual por cada máquina en la que sea instalado [1].

También existe software gratuito, como el caso de *Midas (Microarray Data Analysis System)* [2], implementado en Java, sin embargo, en este sistema no se puede trabajar directamente con el archivo obtenido después de la lectura del microarreglo, ya que requiere de un formato de entrada particular, lo que exigía al usuario modificar manualmente los archivos para poder usar Midas.

La curva de aprendizaje que el usuario necesita para poder utilizar estas herramientas en muchos casos es alta. Esta es la razón principal por la que algunos investigadores optan por usar más de un software para realizar un sólo análisis.

La motivación de este trabajo fue desarrollar una herramienta de software que reuniera varios de los algoritmos existentes para cada una de las operaciones del preprocesamiento de los datos y le permitiera al usuario una interacción con el sistema en el momento de elegir la metodología a utilizar que ofreciera los mejores resultados para los propósitos del investigador de una manera sencilla.

[1] `http://www.silicongenetics.com/cgi/SiG.cgi/index.smf`
[2] `http://www.tigr.org/software/tm4/midas.html`

4.2. Desarrollo

El primer paso de este proyecto consistió en hacer una investigación acerca de los métodos existentes para el análisis de microarreglos, así como de los algoritmos usados en este procedimiento para incluir en el sistema de software a desarrollar aquellos sugeridos en la literatura sobre el tema, con la finalidad de que estos algoritmos cuenten con buena aceptación de los investigadores que harían uso de esta herramienta.

Uno de los puntos más importantes para el desarrollo del proyecto fue la elección del lenguaje de implementación, ya que se debían aplicar una serie de cálculos estadísticos a grandes cantidades de información y la fiabilidad de los resultados depende en gran medida de la implementación de las operaciones del preprocesamiento.

4.2.1. Elección del lenguaje

Dadas las características del sistema que deseabams implementar era necesario un lenguaje que contara con bibliotecas especializadas para cómputo estadístico con la finalidad de hacer uso de ellas. El lenguaje R fue la opción que se ajustó a estas necesidades, además cuenta con una amplia gama de opciones para la visualización de la información y muchas aplicaciones de distribución libre relacionadas con la bioinformática y en particular con el análisis de los microarreglos están desarrolladas en este lenguaje. La que sigue es la definición de R incluida en la página web de R Project [22].

"R es un conjunto integrado de programas para manipulación de datos, cálculo y gráficos. Entre otras características dispone de:

- almacenamiento y manipulación efectiva de datos,
- operadores para cálculo sobre variables indexadas (Arrays), en particular matrices,
- una amplia, coherente e integrada colección de herramientas para análisis de datos,
- posibilidades gráficas para análisis de datos, que funcionan directamente sobre la pantalla o la impresora, y
- un lenguaje de programación bien desarrollado, simple y efectivo, que incluye condicionales, ciclos, funciones recursivas y posibilidad de entrada y salida."

R provee medios para comunicarse con otros lenguajes que facilitan la implementación de interfaces gráficas como son Java, Tcl/Tk y Gtk. Una descripción más profunda de R se proporciona en el siguiente capítulo.

4.3. Diseño del Sistema

En un principio, se pretendió implementar cada una de las funciones del preprocesamiento, de tal manera que el investigador las ejecutara una a la vez desde el prompt

de R. Sin embargo, este enfoque obligaba al investigador a definir cada una de las estructuras que serían tomadas como parámetros por estas funciones, así como las dimensiones del grid (número de renglones, columnas, meta-renglones y meta-columnas del microarreglo). Además de que el usuario debía aprender las bases del ambiente R, por lo menos su sintaxis. Considerando las complicaciones que esta forma de proceder ocasionaba al usuario, se tomó la decisión de realizar la implementación "orientada a objetos".

El término *orientación a objetos* se refiere, en este caso, a una forma de modularizar las estructuras y tratarlas como objetos de alguna clase con ciertos atributos, es decir, no se está siguiendo el paradigma de orientación a objetos, ya que R es un lenguaje funcional (impuro) y solo se trata de una emulación, lo que se explicará con más detalle en el capítulo siguiente.

El desarrollo del sistema se dividió en dos ciclos, el primero contempló la implementación de cada una de las funciones del análisis para que funcionaran de manera independiente con respecto a las demás y en el segundo ciclo se implementó la interfaz gráfica y se liberó el sistema.

4.4. Ciclo I

El ciclo uno, estuvo subdividido en tres fases:

- Análisis y diseño
- Implementación
- Integración y pruebas.

4.4.1. Análisis y diseño

En esta fase se definieron a grandes rasgos las funciones a implementar, es decir, los nombres de éstas, sus argumentos y la interacción entre ellas; se definió también la estructura de los "objetos" que tomarían el lugar de los argumentos haciendo posible esta interacción. Todo esto con la finalidad de lograr una implementación apegada al paradigma orientado a objetos.

El diseño de genArise consta de dos clases: la clase *Spot* que es una abstracción de los datos del microarreglo (Cy3, Cy5, background de Cy3, background de Cy5 e identificadores) y la clase *DataSet*, cuyos objetos son el resultado de aplicar el preprocesamiento a un objeto de la clase Spot y están listos para realizar el análisis.

En esta fase también se decidió la forma en la que se estructuraría el sistema, de manera que éste fuera lo más flexible, incrementable y adaptable a futuros cambios con

un mínimo esfuerzo. Al final se determinó distribuir las funciones implementadas en distintos archivos:

- classes.R Contiene la definición de las clases.
- input.R Contiene las funciones de entrada de datos de genArise.
- output.R Contiene las funciones de salida de datos de genArise.
- var.R Contiene la definición de las funciones que construyen las variables principales del análisis.
- imageLimma.R Contiene la función para graficar una imagen de colores representando el log(ratio) para cada spot en el arreglo.
- operations.R Contiene la definición de las operaciones del preprocesamiento.
- Zscore.R Contiene la definición de la función que obtiene los valores para su posterior análisis.
- plot.R Contiene la definición de las funciones para la visualización de todas las gráficas.

De esta manera, el sistema cuenta con una mayor facilidad de abstracción, modularización, y la gran cantidad de código queda organizado en trozos manejables. Además de que el programa resulta:

- más corto, porque se puede reutilizar el código.
- más fácil de probar, porque cada función se puede probar por separado.
- más fácil de entender, porque el programa es menos confuso y mejor organizado.
- más confiable, porque se reduce la cantidad de código cuando se reutilizan las funciones, por lo que hay menos oportunidad de cometer errores.

Finalmente, esta estructura permitirá escribir más rápido el código, ya que, por ejemplo, se pueden implementar algunas funciones auxiliares que realicen cálculos estadísticos básicos que puedan ser llamadas dentro de la función principal que regrese el resultado final. O mejor aún, se puede encontrar una buena biblioteca que alguien ya implementó y realize los cálculos necesarios sin que tengamos que implementar todo desde cero.

4.4.2. Implementación

Las clases implementadas en el sistema sólo determinan la estructura de los objetos, es decir, no determinan el comportamiento y el estado, así que no se sigue completamente el paradigma de orientación a objetos, pero facilitan la interacción de las funciones a lo largo del análisis, ya que sus argumentos y su valor de regreso siempre son ejemplares de estas clases. Debido a que la aplicación de las transformaciones se realiza de manera secuencial una función puede recibir como argumento el resultado de alguna otra.

El esquema correspondiente a las clases implementadas se muestran en la Figura 4.1:

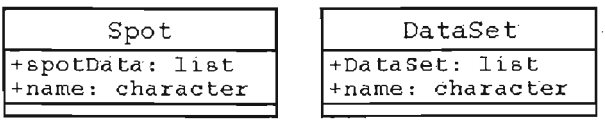

Figura 4.1: Diagrama de clases

Para esta fase, en la implementación de la normalización, se aprovechó una de las bibliotecas con las que cuenta R. Nos referimos al paquete llamado *locfit* que implementa el procedimiento Lowess [23]. El código, desarrollado dentro de los laboratorios Bell, está escrito mayormente en C, pero incluye una interfaz que permite usar locfit como una biblioteca de R. Esto proporciona una mayor confiabilidad a los resultados, debido a que este paquete ya había sido previamente aceptado en el CRAN de R que es un repositorio de funciones para el lenguaje R poniendo especial atención en los paquetes de carácter estadístico.

4.4.3. Integración y pruebas

Las funciones implementadas en el sistema se pueden aplicar de manera independiente, debido a esto, el proceso de integración no se llevó a cabo. Por tanto, esta fase únicamente consistió en una serie de pruebas entre las que destacan:

- Posibles errores durante la ejecución del programa.
- Comparación de los resultados obtenidos contra los obtenidos por Midas (la otra herramienta de software para el análisis de los datos de microarreglos mencionada previamente).
- Manejo de posibles errores cometidos por el usuario.

Errores durante la ejecución

En esta etapa se considera únicamente el caso en que los valores de entrada de alguna función incluida internamente en alguna de las rutinas del análisis puedan producir un valor inválido: NaN (Not a Number) o Inf (Infinito) ocasionando que el programa se detenga. Un ejemplo, es la función logaritmo que se utiliza para calcular el valor del *ratio* ($log2(\frac{Cy3}{Cy5})$). Cuando el valor del cociente de las intensidades es negativo no se puede calcular su logaritmo, así, cuando un elemento cae en este caso, éste es eliminado. No se pierde información, debido a que los valores del cociente debe ser positivo, en otro caso, puede tratarse del resultado de un error durante la fase experimental.

Comparación de resultados

Debido a que Midas es un software utilizado por algunos investigadores para hacer la normalizaciñn, se tomó la decisión de ajustar los resultados de genArise a los de Midas para esta etapa del preprocesamiento como medida de confiabilidad. Se utilizó la función locfit del paquete locfit para este propósito; sin embargo se tuvo que invertir tiempo en encontrar los parámetros correctos para el ajuste de resultados con Midas, como el tamaño del subconjunto de datos para calcular el factor Lowess para cada punto y el grado del polinomio local.

Lowess se basa en la idea de que cualquier función se puede aproximar bien en una vecindad pequeña por un polinomio de orden bajo, por eso casi siempre se utilizan polinomios de primero o segundo orden [24]. En genArise,utilizando un polinomio de grado uno se obtiene un porcentaje final de coincidencias de 92 % con respecto a Midas (utilizando el mismo tamaño del subconjunto de puntos), lo cual fue satisfactorio.

Errores del usuario

Aquí se contemplaron únicamente los errores en la entrada de datos, por ejemplo:

- Formato del archivo incorrecto, es decir, no se encontraba separado por tabuladores.
- Se proporcionaban las columnas incorrectas, introduciendo valores no numéricos en las columnas correspondientes a las señales y al background.
- Archivos incompletos, es decir, omitiendo alguna de las columnas necesarias para el análisis.

4.4.4. Documentación

Para el ciclo I, únicamente se elaboró un manual que incluye el API de genArise, así como la forma en la que cada una de las funciones debe ser utilizadas. El formato de este manual es pdf.

4.5. Diseño de la Interfaz

Para esta etapa se consideraron los criterios ergonómicos para el diseño de interfaces que permitirán facilitar el trabajo tanto de los usuarios principiantes como de los usuarias expertos.

4.5.1. Guía

Este criterio hace referencia a los medios disponibles para guiar a los usuarios a través de su interacción con el sistema. Este criterio está dividido en cuatro subcriterios:

Incitación

En este punto se hace referencia a todos los medios disponibles para llevar a los usuarios a la fabricación de acciones específicas, así como a todos los medios que ayudan a los usuarios a conocer las alternativas posibles que puede seguir dentro del sistema.

Al cumplir con el subcriterio de incitación, se le debe ayudar al usuario a identificar el lugar donde se encuentra, por este motivo las ventanas que conforman la interfaz de genArise aparecen de manera secuencial como ayuda para que el usuario pueda identificar en que paso se encuentra dentro de la secuencia del análisis (figura 4.2).

Figura 4.2: Secuencia en la que aparecen las ventanas dentro del sistema.

Para cumplir también con este objetivo, se colocaron opciones para que el usuario identifique la operación que está aplicando sobre los datos, también se colocaron entradas fácilmente identificadas por el usuario, como menús, botones y entradas de texto, de esta manera se espera que le sirvan de guía al usuario para llevar a cabo cada una de las operaciones.

Agrupación

Este subcriterio hace referencia a la distinción entre los elementos de las ventanas desplegadas. La organización visual de los campos de información debe ser fácilmente identificable. Se deben de tomar en cuenta la topología, distribución y características básicas de los elementos desplegados.

La agrupación que se realiza en el sistema se basa en criterios de agrupación y distinción por localización. Por ejemplo, en la ventana principal se encuentran agrupados por un lado los datos correspondientes al archivo a analizar, en otro agrupamiento se encuentran las opciones para graficar y finalmente en la parte inferior se localiza un área de texto no editable en donde el sistema le indica al usuario qué operaciones ha realizado a lo largo del análisis (Figura 4.3).

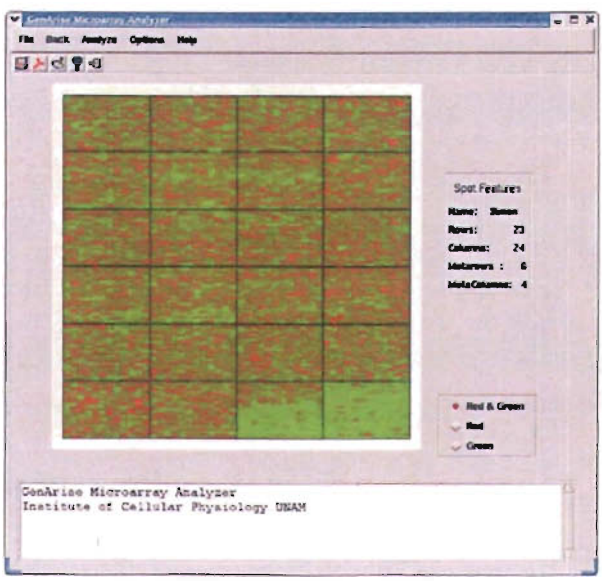

Figura 4.3: Agrupación en la ventana principal.

Retroalimentación inmediata

Este subcriterio se refiere a las respuestas que el sistema brinda a cada acción del usuario. Para cumplir con esta característica se verificó que cada vez que el usuario seleccionara una de las opciones del sistema se ejecutara la opción correspondiente. Cuando las operaciones que se deben llevar a cabo ocupan una cantidad de tiempo que haga suponer al usuario que la acción solicitada no tuvo efecto, el apuntador del mouse cambia para que el usuario sepa que el sistema realmente esta trabajando.

Legibilidad

Este criterio se refiere a las características que tiene la información presentada en pantalla, por este motivo se prestó atención al tamaño y figuras utilizadas, para que faciliten la lectura al usuario.

4.5.2. Carga de trabajo

Este criterio concierne a todos los elementos de la interfaz que juegan un papel en la reducción de la carga perceptual y cognoscitiva del usuario. Este criterio también se divide en subcriterios.

Brevedad

Se refiere a la carga de trabajo perceptual y cognoscitiva para las entradas y salidas individuales, y para un conjunto de entradas; es decir, el conjunto de acciones necesarias para realizar una meta o tarea. La brevedad corresponde a la meta de aminorar la lectura y la entrada de la carga de trabajo y el número de acciones a seguir.

Para responder a este criterio, cada una de las ventanas que se van desplegando a lo largo de la utilización del sistema, contienen únicamente los elementos relacionados con el propósito general de la ventana. Es decir, en la entrada de los datos de inicio del programa todos los campos que el usuario tiene que llenar están contenidos en ella para no darle al usuario diferentes ventanas para cada campo, lo que implica más carga de trabajo. De la misma manera, el seguir con este criterio se ha logrado una buena prevención de errores que podían ser cometidos por el usuario.

Dentro de este criterio se pretende minimizar el número de acciones necesarias para completar una tarea por parte del usuario, a esto se le denomina carga de trabajo y se pretende que ésta no sea tediosa para el usuario, buscando aminorar lo más posible los pasos que el usuario realiza en una tarea. Por esta razón, las pantallas desplegadas por el sistema sólo contienen información correspondiente a un proceso en el análisis (concisión).

Densidad de la información

Concierne a la carga de trabajo del usuario desde un punto de vista perceptual y cognoscitivo ocasionada por los grupos de elementos y no por elementos aislados como en el caso de la brevedad. El sistema no muestra información que no es necesaria. De hecho en el sistema solo se muestran las opciones que tiene el usuario, con esto la cantidad de información presentada en las pantallas no es excesiva.

4.5.3. Control explícito

Hace referencia al procesamiento por parte del sistema de acciones explícitas del usuario, así como el control que debe tener el usuario sobre el proceso.

Acciones explícitas

Se refiere a las relaciones entre el procesamiento de la computadora y las acciones de los usuarios. Esta relación debe ser explícita; es decir, se deben procesar únicamente aquellas acciones solicitadas por el usuario y solo cuando éste así lo requiera.

Para cumplir con este criterio se buscó que cada vez que el usuario seleccionara una operación, ésta se aplicara de manera inmediata y el sistema no podrá realizar ni se le podrá hacer otra petición hasta que haya terminado de realizar la operación requerida inicialmente.

Control de usuario

Se refiere al hecho de que los usuarios siempre tendrán el control del procesamiento del sistema. Para esto, en genArise se muestras opciones apropiadas cuando el usuario hace ciertas peticiones, entre estas opciones se encuentran quellas que le brindan al usuaerio cierto control sobre el procesamiento del sistema (Figura4.4).

Hay que destacar que el control de usuario está limitado, debido a que las operaciones seleccionadas se ejecutan rápidamente, por lo que anticipar las acciones a ejecutar pueden originar una mayor carga de trabajo; sin embargo, la posibilidad de deshacer las transformaciones ejecutadas es una de las opciones de genArise, lo que le permite al usuario seguir teniendo control sobre el sistema.

4.5.4. Adaptabilidad

La adaptabilidad de un sistema se refiere a su capacidad para comportarse de manera contextual y de acuerdo a las necesidades y preferencias del usuario. En el caso de

Figura 4.4: Control de usuario.

genArise, este criterio no lo pudimos aplicar del todo, ya que el único cambio de contexto que podemos encontrar en el sistema es cada despliegue de las distintas gráficas.

4.5.5. Manejo de errores

Se refiere a los medios disponibles para prevenir o reducir errores y recuperarlos cuando estos suceden. En genArise, lo errores que en este contexto se contemplan son: entrada de datos inválidos o formatos inválidos en la entrada de datos, a los que ya nos referimos anteriormente.

Protección contra errores

Hace referencia a los medios disponibles para detectar y prevenir errores. En el sistema se hace una prevención de muchos errores que pueden ser cometidos por el usuario, principalmente en inhabilitar opciones que no deben ser seleccionadas. Esta forma de proceder se realiza en todas las ventanas del sistema, y se inhabilitan para que el usuario identifique que opciones tiene disponibles en cada paso del análisis.

Calidad de los mensajes de error

Debido a la prevención de los errores, la cantidad de los mensajes de error se ven disminuidos pero se pretendió que tuvieran un mensaje claro del error que ocurrió para que el usuario lo identifique.

Corrección de errores

Se refiere a los medios disponibles para que los usuarios corrijan sus errores. En el caso de genArise, el mensaje aparece antes de que se haya ejecutado el error, el usuario debe volver a repetir la acción, para ello se optó por ùn mensaje de error claro (Figura 4.5).

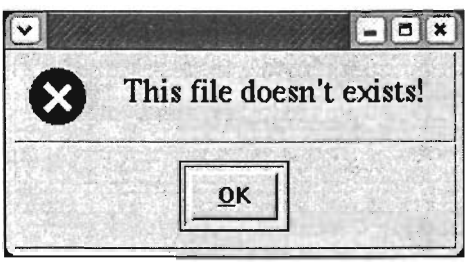

Figura 4.5: Mensaje de error.

4.5.6. Consistencia

Se refiere a la manera en que el diseño de la interfaz mantiene una estructura similar cuando la secuencia de aparición de las ventanas respnden a operaciones dentro del mismo contexto. Por otro lado, si existe un cambio de contexto, la respuesta que muestre la interfaz debe diferenciarse de lo que mostraba anteriormente.

Todas las ventanas del sistema tienen la misma estructura, al igual que los mensajes que se le muestran al usuario, esto es con respecto a la ubicación de los botones, las opciones y las gráficas.

4.5.7. Significado de códigos

Califica la relación entre un término y/o un signo, y el objeto o comando al que hace referencia. Los códigos y nombres son importantes para los usuarios cuando existe una relación clara entre tales códigos y acciones.

En el diseño de la interfaz se optó por utilizar imágenes fácilmente identificables por el usuario. En particular, optamos por incluir íconos semejantes a los utilizados por otros sistemas de uso común, de tal manera que el usuario ya conoce la relación que existe entre cada ícono y la operación que se ejecutará (Figura 4.6).

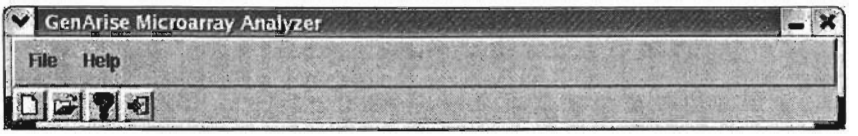

Figura 4.6: Imágenes en el menú principal.

4.6. Ciclo II

Para este segundo ciclo se propusieron los siguientes objetivos:

1. Desarrollo de la interfaz gráfica.
2. Integración del sistema como paquete de R.
3. Liberación del paquete

Para conseguir estos objetivos, este segundo ciclo se estructuró en las mismas tres fases del ciclo I:

- Análisis y diseño
- Implementación
- Integración y pruebas.

De esta manera, la idea era adaptar la parte del software que quedó lista en el primer ciclo para agregarle la GUI y poder liberar el software a la comunidad interesada.

4.6.1. Análisis y diseño

Una vez concluido el diseño de la interfaz gráfica para genArise, se procedió a planear la forma en la que ésta debía de implementarse. Para la interfaz gráfica cumpliera con todos los requerimientos del sistema y la determinaciń final contemplabase determinóusar la biblioteca *Tcl/Tk*.

Tomando en cuenta la estructura que tenía el sistema, la implementación de la interfaz gráfica debía de seguir la misma estructura, de manera que a los archivos que contenían el código del ciclo I no se les tuviera que modificar nada y fueran totalmente independientes de la interfaz, dando como resultado una arquitectura de tres capas estructurada de la siguiente manera:

Capa de Presentación. Reúne todos los aspectos del software que tiene que ver con las interfaces y la interacción con los diferentes tipos de usuarios. Estos aspectos típicamente incluyen el manejo y aspecto de las ventanas, el formato de los reportes, los menús y las gráficas en general.

Capa del Dominio de la Aplicación. Reúne todos los aspectos del software que automatizan o apoyan los procesos de *negocio* que llevan a cabo los usuarios. Estos aspectos típicamente incluyen las tareas que forman parte de los procesos, las reglas y restricciones que aplican. Esta capa también recibe el nombre de la capa de la Lógica de la Aplicación.

Capa de Datos. Reúne todos los aspectos del software que tienen que ver con el manejo de los datos persistentes, por lo que también se le denomina la capa de las Bases de Datos, entendiéndose como base de datos en este caso, a los archivos con los resultados del experimento.

Viéndolo de esta manera, la labor que quedó pendiente en el ciclo I era la capa de presentación. Las funciones de esta capa se implementaron en los siguientes archivos:

- applications.R Contiene funciones de *miscelánea* para distintos objetivos de la interfaz.
- baseGUI.R Contiene la implementación de las ventanas para cada etapa del análisis.
- base.R Contiene todas las variables que almacenan la información necesaria para el proyecto.
- genArise.R Contiene la función que inicializa la ventana principal del sistema.
- help.R Contiene la implementación de la ventana de ayuda de genArise.
- openProyect.R Contiene las ventanas para abrir un proyecto tiempo después de haberlo realizado.
- postAnalysis.R Contiene la implementación de las operaciones de conjuntos entre diferentes análisis.

Por otro lado, el diagrama de estructura de la Figura 4.7 muestra el conjunto de archivos, con sus respectivas funciones y objetos importantes que forman parte del sistema, junto con las relaciones existentes entre estas funciones y objetos en el modelo de la capa del dominio de la aplicación y la capa de datos.

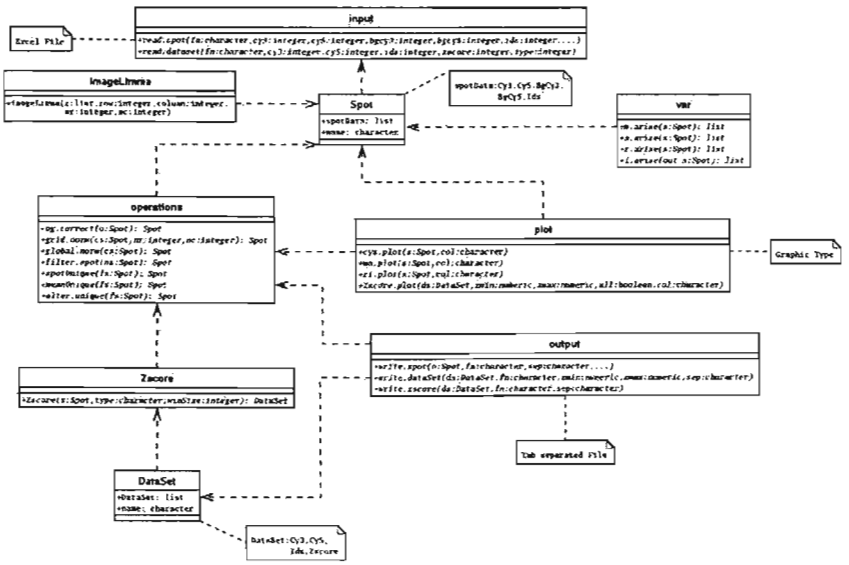

Figura 4.7: Diagrama de estructura

4.6.2. Implementación

En un principio el sistema fue pensado únicamente para Linux; sin embargo, debido a que Windows es la plataforma comúnmente utilizada por los usuarios, se determinó que implementación de la GUI debía utilizar bibliotecas que permitieran la portabilidad.

Debido a esto, se usó la biblioteca *Tcl/Tk* que es compatible con una gran cantidad de plataformas, entre ellas: Windows y Linux, además de contar con una gran cantidad de componentes gráficos (widgets). Con esta biblioteca, para el despliegue de las gráficas en la GUI fue necesario el uso de la biblioteca *tkrplot* que colocaba las gráficas de R en una imagen de Tk lo cual eliminaba cualquier problema con los dispositivos gráficos.

El paquete base de R ofrece una interfaz con Tcl/Tk, pero habían algunas características de genArise que no venían incluidas en el API de esta interfaz, así que fue necesario incluir algunos requerimientos extras. En particular, genArise usa widgets del paquete BWidget y del paquete TkImg, ninguno de los cuales viene incluido con la instalación base de R. Para cubrir esta limitación, si se usa Windows, es necesario instalar ActiveTcl 8.4.x que es la instalación completa de Tcl. Por otro lado, si la plataforma utilizada es Linux, se deben instalar los rpms para Tcl y Tk 8.3 o posterior.

4.6.3. Integración y Pruebas

Para esta esta fase, se debió integrar la capa de presentación con las otras dos capas y esto no fue muy complicado, ya que sólo haciamos uso de los resultados obtenidos mediante la ejecución de las funciones de la lógica del programa para desplegar las gráficas en el dispositivo gráfico de Tcl/Tk, así, cada uno de los menús de genArise incluyen dentro de su funcionalidad una llamada a alguna de las funciones de la capa lógica del sistema o de la capa de datos.

La etapa de pruebas de esta fase se tomaron en cuenta los errores más comunes que podría cometer un usuario al momento de introducir los datos como por ejemplo, indicar las columnas incorrectas de los datos, proporcionar el nombre de un archivo inexistente, seleccionar incorrectamente el origen del experimento, entre otros.

Los posibles errores que se podrían producir durante el análisis se manejaron mediante la desactivación de los menús de las operaciones cuando éstas implicaban un orden que no era válido para el análisis; por ejemplo: cuando se aplica primero el filtrado por intensidades, el menú de la normalización por grid debe ser desactivado, pues no es posible aplicar la normalización por grid una vez filtrado el microarreglo. Así que este tipo de errores no es posible que se produzcan.

4.6.4. Liberación del paquete

Una vez que genArise se ejecutó con éxito tanto en Linux como en Windows, se determinó hacer de genArise una biblioteca para compartirla con la comunidad dedicada al análisis de microarreglos por medio de un sitio especializado en herramientas para la bioinformática implementadas en R, el cual exige ciertos requisitos entre los cuales destacan[3]:

- Superar un proceso de verificación del código sin que sean detectados *errores* o *warnings*
- El paquete debe poder ejecutarse en la última versión liberada de R.
- El paquete debe contener un directorio *inst/docs* que incluirá la documentación en LaTeXpara generarla posteriormente en diferentes formatos.

4.6.5. Documentación

Para este segundo ciclo, fue necesario modificar el manual del API de genArise elaborado en el ciclo previo, para incluir algunas de las funciones derivadas de la capa de

[3] http://bioconductor.org/faq.html

presentación. Sin embargo, también fue necesario elaborar un manual de usuario tanto para la parte de genArise desde la línea de comandos, como para su interfaz gráfica.

Pensando en los alcances que podría tener la distribución de genArise estando en Bioconductor, se decidió traducir el manual de usuario al inglés, además de elaborar otros manuales más específicos como un manual de instalación para Windows.

Capítulo 5

R: Especificaciones del Lenguaje

R es un sistema para cómputo estadístico y visualización de gráficas. Entre otras cosas, contiene un bien desarrollado, simple, pero efectivo lenguaje de programación, así como interfaces con otros lenguajes y facilidades para la corrección de errores.

R está basado en el lenguaje S desarrollado en los laboratorios Bell (pertenecientes a AT&T) en los ochentas, por esto se dice que R es un dialecto de este lenguaje. El principal diseñador de S, John M. Chambers, se hizo acreedor en 1998 al premio de la ACM (Association of Computer Machinery) en Sistemas de Software gracias a S, que "ha alterado para siempre la forma en como la gente analiza, visualiza y manipula datos ... "

En pocas palabras, R es un ambiente interactivo para análisis de datos, cómputo estadístico y visualización distribuido de manera libre y es el resultado de un activo movimiento entre estadísticos en la búsqueda de un ambiente *robusto, portable, libre* y que además fuera *programable*, capaz de aplicarse en la resolución de pròblemas complejos así como en el análisis de datos.

5.1. CRAN

Como se mencionó anteriormente, R implementa muchas técnicas estadísticas y las incluye en su entorno base en forma de bibliotecas, pero también existe una gran cantidad de estas bibliotecas que son implementadas por gente que contribuye al proyecto R y son distribuidas en forma de bibliotecas o paquetes (*packages*), siendo un medio práctico para empaquetar y reutilizar componentes, lo que le da a R una gran capacidad de extendibilidad. La distribución de estas bibliotecas se realiza por medio del **CRAN**.

El CRAN por sus siglas en inglés "Comprehensive R Archive Network" se trata de un conjunto de mirrors (sitios web con idéntico contenido) con los fuentes y los binarios de la distribución más reciente de R para varios sistemas operativos como son (Linux, Mac OS Classic, Mac OS X y MS Windows), las extensiones o contribuciones en forma de bibliotecas y la documentación de R, así como listas de correo y un sistema para reportar Bugs.

El principal servidor del CRAN se encuentra en la Universidad Técnica de Viena (TU Wien), Austria, y la dirección es:

```
http://cran.R-project.org/
```

Los mirrors son muchos a lo largo del WWW y basta consultar la dirección

`http://cran.R-project.org/mirrors.html` para obtener una lista de mirrors completa y actualizada.

La sintaxis del lenguaje R recuerda un poco a C, pero la semántica es del tipo de los lenguajes de programación funcional (si bien impuro, ya que, por ejemplo, sus estructuras de datos no son inmutables) con grandes afinidades con Lisp y Scheme.

El propósito de este capítulo es presentar las principales características del lenguaje, que son conceptos importantes cuando se programa en R y para entender la implementación de genArise. Esto es, los objetos con los que trabaja, los detalles del proceso de evaluación de expresiones y de los argumentos de las funciones, el sistema de tipos, las interfaces con otros lenguajes como C y Fortran.

5.2. Lenguajes: Interpretados y Compilados

R es un lenguaje interpretado, lo que significa que cada expresión individual es leída e inmediatamente evaluada por el intérprete que está en un constante ciclo a la espera de la siguiente expresión. A este ciclo se le conoce como *read-eval-print loop*. De esta manera, el intérprete es el programa que está entre las funciones escritas en R y la máquina sobre la que esas funciones se ejecutarán. Sin embargo, cualquier programa interpretado necesita del intérprete asociado para poder funcionar.

Otro tipo de lenguajes, por ejemplo C, son lenguajes compilados. Los programas implementados en este tipo de lenguajes son por completo traducidos por un compilador al lenguaje de la máquina sobre la que va a correr el programa. Una vez hecho esto, el programa compilado funcionará independientemente del compilador, aunque el código de máquina generado será específico de la plataforma.

Una de las ventajas que proporcionan los lenguajes interpretados está el hecho de que permiten la implementación incremental (podemos implementar una a una las funciones que en conjunto formarán nuestra aplicación), por lo que en muchas ocasiones

es utilizado para realizar prototipos. Después de evaluar un prototipo se puede determinar si alguna función o incluso alguna porción de este debería ser codificada en C o incluso en Fortran para hacerla más eficiente y posteriormente se incorpora este código compilado al resto de la aplicación.

La desventaja de un lenguaje interpretado es que requiere de una mayor cantidad de recursos dado que estos se comparten entre la ejecución del programa y el intérprete. En cambio, un programa compilado requiere menos memoria porque al contar con el código completo del programa, el compilador puede realizar una fase de optimización para ejecutar las tareas de la manera más eficiente [25].

5.3. Especificaciones del Lenguaje

Cuando una persona u organización comienza a crear algo teniendo un modelo a seguir, lo hace con la idea de que el resultado final de sus esfuerzos superen las características del modelo utilizado. De esta forma comenzó en 1994, por Ross Ihaka y Robert Gentleman del Departamento de Estadística de la Universidad de Auckland en Nueva Zelanda, el **diseño** e **implementación** de un nuevo lenguaje para cómputo estadístico. El modelo que tenían como referencia y que influenció en gran medida su trabajo consistía en dos lenguajes ya existentes: S y la programación funcional, en particular Scheme un dialecto de LISP [26].

La intención era reunir las ventajas de ambos lenguajes y combinarlas. ¿El resultado? Un lenguaje cuya sintaxis es muy similar a la de S pero con su implementación y semántica basada mayormente en Scheme [1].

En esta sección se tratarán algunos elementos que se constituyen como piezas fundamentales para hacer de R un lenguaje sencillo pero con características excepcionales para obtener resultados confiables cuando se realizan operaciones sobre grandes cantidades de datos.

5.3.1. Objetos

En todos los lenguajes de programación, las variables son el medio para acceder a los datos almacenados en memoria. En R, no es posible acceder directamente a la memoria, en cambio, proporciona varias estructuras de datos especializadas conocidas como **objetos** [26]. Podemos hacer referencia a un objeto a través de los **símbolos** (variables).

Casi todo en R es un objeto; sin embargo, podemos mencionar a los siguientes como los objetos más importantes:

[1] El alcance léxico y closures vienen directamente de Scheme

- Vectores. Colección ordenada elementos del mismo tipo.
- Arreglos.Generalización multidimensional de vector. Elementos del mismo tipo.
- Data Frames. Como los arreglos, pero permiten elementos (columnas) de distintos tipos. El objeto más habitual para manejo de datos procedentes de experimentos.
- Listas. Un "vector generalizado". Cada lista está formada por componentes (que pueden ser otras listas), y cada componente puede ser de un tipo distinto.
- Funciones. Código.

5.3.2. Sintaxis

La definición de un lenguaje de programación consiste en describir la *sintaxis* del lenguaje (el aspecto de sus programas) y la *semántica* de éste (el significado de estos programas). Con la sintaxis se especifica la *forma* en la que un usuario del lenguaje puede expresarse y esta depende exclusivamente del diseño del lenguaje. A partir de esta definición de sintaxis es obligatorio describir también cada uno de los elementos con los que se construyen desde los átomos del lenguaje (elementos atómicos) hasta las frases (o expresiones) más complicadas. [27]

Dentro de R, cualquier expresión completa debe caer dentro de una de las clases que se especifican en la tabla 5.1.

Clase	Expresión
Literales	Se trata de los objetos más simples que reconoce el lenguaje. Números, cadenas de caracteres o nombres de variables son literales.
Llamadas a funciones	Una de las expresiones de mayor utilidad en R. Una llamada puede ser cualquier uso de una función con los argumentos correctos o un operador.
Asignaciones	Asocian el nombre de una variable con un valor (denotación).
Condicionales	Permiten la ejecución condicional de una expresión dependiendo del valor lógico de una condición definida.
Control de flujo	Dirigen la evaluación ya sea fuera del ciclo o a la siguiente iteración.

Tabla 5.1: Clases de expresiones válidas en R.

Una expresión completa puede estar constituida por muchas expresiones (sub-expresiones) de manera inductiva.

5.3.3. Evaluación de expresiones (Alcance Léxico)

Una expresión de cualquier lenguaje no es suficiente en sí misma para poder obtener el valor que será regresado. Es necesario además, asociar de alguna forma los valores con los nombres de variables (símbolos en R) que se encuentran en esta expresión.

Como sabemos, una variable puede aparecer en dos contextos diferentes: como *referencia* cuando se hace uso de la variable o como *declaración* cuando se introduce una variable como nombre para un valor. Al valor nombrado por una variable se le suele llamar la *denotación* de esa variable. Esta denotación se obtiene de alguna declaración y se dice que la variable se refiere a esta declaración.

Una variable debe designar un lugar donde guardar sus valores. Estos lugares a su vez deben ser almacenados en alguna estructura para el proceso de evaluación. Nos referiremos a estas estructuras como *ambientes* y se dice que una expresión es evaluada en un ambiente.

Un ambiente consiste en una lista de *frames*, los cuales podemos imaginar como una tabla de asociaciones que asocia el nombre de la variable con su correspondiente valor (un frame debe contener a lo más una asociación para cualquier variable). Un valor ligado a una variable en un frame anterior en la secuencia tomará precedencia sobre un valor ligado a la misma variable en un frame posterior en la secuencia. Se dice que el primer valor *ensombrece* o *enmascara* al segundo.

Cuando una variable no tiene asociado un valor en un frame en particular, esta variable se conoce como una variable libre en ese frame. Por ejemplo, cuando la función $f \rightarrow function\ (x)\ x\ *\ (a/2)$ es llamada como f(2), x es un parámetro formal y está *ligada* al valor 2 en el frame local de la llamada a la función, pero a es una variable libre en ese frame.

¿Pero en qué momento se crean los ambientes? En R, el ambiente de una función (el ambiente creado por una llamada a la función) comprende el frame local de la llamada a la función, así como el ambiente en el que la función fué definida. Esta regla se conoce como alcance léxico [2].

Cada lenguaje de programación debe contar con reglas para determinar la declaración a la cual cada variable hace referencia. En esta sección se discute cómo estas reglas son manejadas en R, así como los efectos que estas tienen sobre la evaluación.

Para ilustrar los conceptos anteriores daremos el siguiente ejemplo:

```
> f <- function (x) x + a
> a <- 10
> x <- 5
```

[2] En la programación funcional se habla de *closures* y los definen como una función junto con un ambiente y algunas veces se suele decir que una función es *cerrada sobre* o *cerrada en* su ambiente de creación.

```
> f(2)
[1] 12
```

Cuando llamamos a la función f, la asociación local de x → 2, *ensombrece* la asociación global de x → 5. La variable a es una variable libre en el frame de la llamada de la función, así que la asociación global a → 10 es la que se toma para la evaluación. Este ejemplo se muestra en el diagrama de la Figura 5.1.

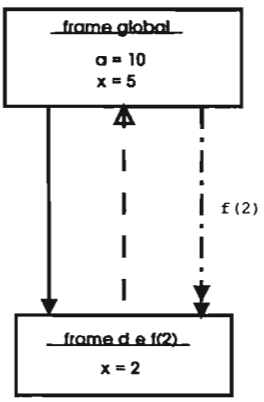

Figura 5.1: Ambiente de la llamada de la función f(2). Cada caja representa un frame, con la asociación de las variables y sus valores en el frame mostrada dentro de la caja. La flecha sólida representa una definición de función: La función f está definida en el frame global. La flecha interrumpida representa la secuencia de frames que comprende el ambiente de la llamada de la función. La variable a, libre en el frame de la llamada, se localiza en el frame global, mientras que la asociación local de x ensombrece la asignación de x en el frame global. La flecha línea-punto representa la llamada. Modificada de [28]

Una situación muy frecuente cuando se está programando es llamar a una función dentro de otra y este es un buen ejemplo para entender las reglas de asociación de R y tratar de compararlo con las de S-PLUS. Así que veremos el comportamiento de ambos lenguajes dado el mismo ejemplo.

```
> f <- function (x) {
+ a <- 5
+ g(x)
+ }
> g <- function(y) y + a
> f(2)
[1] 12
```

En este ejemplo, R y S-PLUS producen el mismo resultado; sin embargo, la justificación es distinta en ambos casos:

- En R, la asociación global a → 10 se usa cuando f llama a g, porque a es libre en g, entonces R continúa la búsqueda del valor asociado con a en el frame de la definición de la función, que en este caso es el frame global.
- En S-PLUS, la asociación global a → 10 se usa simplemente porque a es libre en g, y las reglas de asociación de S-PLUS establecen que se debe buscar en el frame global después del frame local de la llamada a la función.

De esta manera, ambas reglas ignorarán la asociación a → 5 en el frame local de f (Figura 5.2).

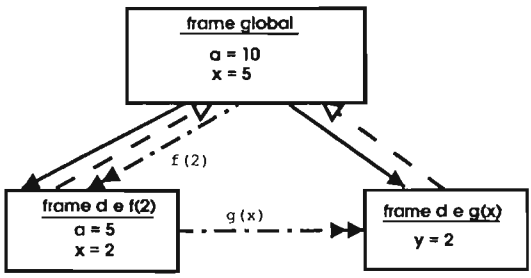

Figura 5.2: En este caso, tanto f como g están definidas en el frame global; f es llamada desde el frame global, mientras g es llamada desde f. La variable libre a en g obtiene su valor desde el frame global tanto en R como en S-PLUS. Modificada de [28]

Si la función g hubiera sido definida dentro de la función f y no en el frame global, si notaremos claramente las diferencias entre el modelo de evaluación de R y el de S-PLUS. Primero veamos como procede R con este ejemplo.

```
> f <- function (x) {
+ a <- 5
+ g <- function (y) y + a
+ g(x)
+ }
> f(2)
[1] 7
```

El ambiente de g comprende el frame local de g seguido del ambiente de f. Como a es una variable libre en g, el intérprete busca el valor de a en el frame local de f; allí encuentra que a → 5, esto ensombrece la asociación global a → 10 y es el valor que se asocia con a (Figura 5.3).

En contraste, en S-PLUS obtendríamos un resultado distinto al obtenido con R.

```
> f <- function (x) {
```

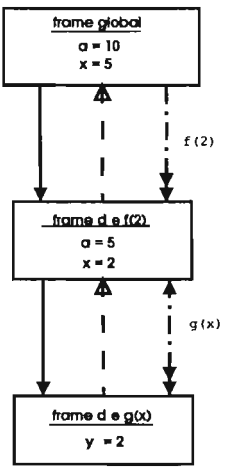

Figura 5.3: Alcance léxico en R: En este caso, tanto f como g están definidas en el frame global; f es llamada desde el frame global, mientras g es llamada desde f. La variable libre a en g obtiene su valor desde el frame global tanto en R como en S-PLUS. Modificada de [28]

```
+ a <- 5
+ g <- function (y) y + a
+ g(x)
+ }
> f(2)
[1] 12
```

Cuando encuentra la variable libre a en g dentro del frame local, el intérprete de S-PLUS ignora la asociación a → 5 en f y busca directamente en el frame global, donde encuentra que a → 10 (Figura 5.4).

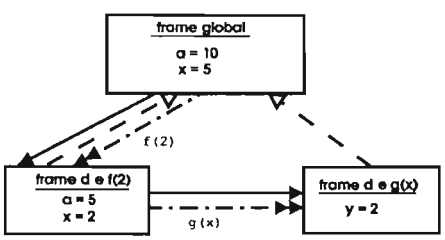

Figura 5.4: Alcance en S-PLUS: Aunque g es una función local definida en f y llamada desde f, la variable libre a en g obtiene su valor de la asignación de a en el frame global. Modificada de [28]

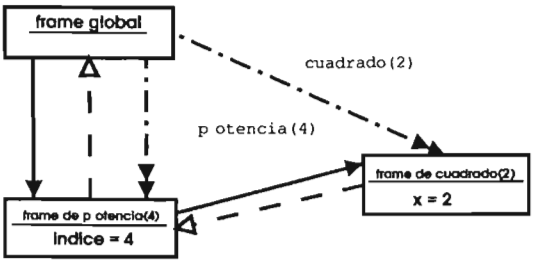

Figura 5.5: Alcance Léxico en R: La función cuadrado está definida en el frame de potencia(4) y así, su ambiente incluye este frame. La variable indice es libre en el frame de cuadrado(2), pero la asociación indice → 2 se encuentra en el frame de potencia(4), aunque cuadrado es llamado desde el frame global. Modificada de [28]

Como en R las variables definidas localmente son visibles a las funciones locales, el alcance léxico de R es más conveniente que las reglas de evaluación empleadas en S-PLUS. Además, el alcance léxico es más poderoso en ciertas circunstancias. Como ejemplo veamos la siguiente función de R:

```
> potencia <- function(indice) {
+ function(x) x^indice
+ }
```

La función potencia regresa un closure (la función junto con su respectivo ambiente) como resultado.

```
> cuadrado <- potencia(4)
> cuadrado
function(x) x^indice
<environment: 0x8c163c8>
> cuadrado(2)
[1] 16
```

Cuando *potencia* es llamada con 4 como argumento, este valor es asociado a la variable local *indice*. Así, la función que es regresada se define en el frame local de la llamada a potencia y *heredà* el ambiente de esta llamada, incluyendo el valor de indice (ver Figura 5.5).

Como S-PLUS no utiliza alcance léxico, este procedimiento falla, pues no conoce el ambiente asociado a la función, por lo que no encontrará un valor asociado a la variable *indice* en la función cuadrado y el intérprete buscará esta asociación en el ambiente global; que en general producirá errores o resultados indeseables.

5.3.4. Evaluación Perezosa

Para la ejecución de un programa en R el intérprete procede a calcular el valor asociado a los argumentos formales dados para las funciones en el programa. As'ı, para calcular el valor de f(t1,...,tn) puede o no ser necesario calcular primero los valores de los argumentos t1, ..., tn.

Ejemplo:

1. Para calcular el valor del dato de entrada num1 + num2 es preciso saber el valor de ambos argumentos, num1 y num2 en cualquier caso.
2. Para calcular el valor del dato de entrada 0 * num no es necesario evaluar num, pues cualquier valor multiplicado por 0 es 0.

En los lenguajes de programación existen fundamentalmente dos estrategias para llevar a cabo la evaluación de los argumentos:

- La estrategia *voraz o ansiosa* (eager), que es usada por Scheme, evalúa todos los argumentos de la llamada a la función antes de evaluar el cuerpo de la función.
- La estrategia *perezosa* (lazy), usada por S-PLUS, sólo evalúa las expresiones dadas como argumentos de la llamada a la función si es necesario hacerlo para evaluar la función.

En este aspecto, R tomo el modelo de evaluación de argumentos de S, es decir, utiliza evaluación perezosa. Y así es como lo hace: en lugar de evaluar las expresiones dadas como argumentos de la función antes de que ocurra la llamada a la función, estas expresiones son *almacenadas* en una estructura de datos conocida como *promesa* junto con un apuntador al ambiente desde el cual fue llamada la función y esta es la estructura que es pasada a la función. Así, la evaluación se realiza únicamente cuando el valor del argumento es requerido y en algunos casos los argumentos nunca serán evaluados. Por eso no se deben usar argumentos a funciones que causen efectos laterales porque no hay garantía de que los argumentos serán evaluados. [29]

Por ejemplo:

```
> bad.function <- function(x, y) {
+  if(y < 10) print(x)
+  else print(y)
+ }
> bad.function(z <- 3, 14)
[1] 14
> z
Error: Object "z" not found
```

```
> bad.function(z <- 3, 4)
[1] 3
> z
[1] 3
```

Entonces, cuando se hacen asignaciones en una llamada a función, puede resultar en un comportamiento inesperado, como el que muestra el ejemplo anterior.

La asignación $z \leftarrow 3$ se hace en el ambiente desde el cual fue llamada la función, pero el comportamiento entre las dos llamadas a la función es distinto debido a la evaluación perezosa. En el primer caso, como y es mayor que 10, la x nunca es evaluada, por lo que la asignación nunca se realiza. En la segunda llamada, al evaluar el cuerpo de la función se cumple la condición del if y, por lo tanto, x es evaluado y la asignación se realiza.

La semántica del paso de parámetros cuando se invoca una función en R es *paso por valor*. En general, podemos pensar en los argumentos de una función como variables locales inicializadas con el valor proporcionado en los argumentos actuales y el nombre del correspondiente argumento formal. Cualquier cambio que se realice dentro de la función en el argumento enviado, no afectará al valor original de las variables utilizadas en el frame de la llamada. Esto es, lo que realmente se está pasando a la función es una *promesa* que encapsula la expresión usada en la llamada.

5.3.5. Sistema de Tipos

Los tipos se utilizan para distinguir un grupo de objetos de otro. Podemos entonces decir que un tipo "x" es un conjunto, y que los elementos de "x" son de "tipo x" . Por ejemplo, el valor "3" es un elemento del conjunto de los enteros, y por lo tanto "3" es de "tipo entero". Esta es la interpretación que generalmente se hace de los tipos en relación con los lenguajes de programación. Denotaremos el hecho de que un valor "a" tiene un tipo "t" escribiendo "a:t".

En general, los lenguajes de programación presentan una serie de tipos predeterminados que llamaremos primitivos. Por ejemplo, en el lenguaje de programación C existen, entre otros, los tipos primitivos: entero y doble o real.

A partir de los tipos primitivos se pueden construir otros, como son los de las funciones, los procedimientos y los pares o n-adas. Si una función f espera un parámetro de tipo "t" y el resultado es de tipo "s", denotamos el tipo de f como "t $\rightarrow$ s". En el caso de un par, si "a:t" y "b:s", entonces "(a, b):t*s". Como un ejemplo, si el operador "+" denota una función con dos parámetros de tipo entero y genera o regresa un valor de tipo entero, entonces podemos escribir "+:entero * entero $\rightarrow$ entero".

Un sistema de tipos consiste en una serie de reglas que restringen el uso de los elementos de cada uno de los tipos. Un ejemplo de una regla es: Si la operación "+"

denota la suma de números enteros, entonces no podemos usar "+" más que para sumar números enteros. La regla general dice que si una función f tiene un tipo "s → t", entonces la función f sólo puede aplicarse a valores de tipo "s", y sólo regresa valores de tipo "t".

Los sistemas de tipos son utilizados en los lenguajes de programación para asegurar que las funciones se apliquen de manera coherente, esto es, que no se intente aplicar una función que opera sobre valores de tipo "t" a un valor de tipo "s", donde "t" es distinto de "s".

Existen, por supuesto, variaciones sobre estas reglas. Por ejemplo, en R se permiten muchas operaciones sin considerar si ellas están definidas para los tipos de los operandos en cuestión. A sistemas de tipos como el de R se les llama *débiles* y se dice que es un lenguaje débilmente tipado.

R es un lenguaje tipado dinámicamente, es decir, el tipo lo adquiere el objeto al momento de la asignación y puede cambiar de tipo durante la ejecución si es que se le asigna un valor de distinto tipo, es decir, el sistema determina el tipo de las variables según su contexto y ese tipo puede ser recalculado si el contexto cambia, pero no se pueden operar de manera inconsistente (por ejemplo, no se puede sumar un valor numérico con una cadena).

Estas características proveén de una gran flexibilidad a los programas pero tienen dos desventajas. La primera es que el compilador o intérprete tiene que insertar, o ejecutar código extra para realizar la revisión de tipos en tiempo de ejecución, lo que hace que la misma sea ineficiente. La segunda desventaja es que en un programa largo podría haber errores de tipos en alguna parte del código, los cuales podrían aparecer cuando el programa esté siendo utilizado por el usuario final, y así provocar problemas serios a éste. Esto obliga al programador a verificar por su cuenta que los objetos con los que se va a operar sean de tipos similares.

En vez de centrar su atención en el tipo de los objetos tan pronto como sea posible (como lo hacen C++ y Java), R (al igual que otros lenguajes débilmente tipados como Python) se concentra lo menos posible en el tipo, y evalúa los tipos solamente si es necesario. Es decir, es posible enviar cualquier mensaje a cualquier objeto, y el lenguaje cuida solamente que el objeto puede aceptar el mensaje (no requiere que el objeto sea de un tipo en particular, como lo hacen Java y C++) [30].

5.3.6. Operaciones Vectorizadas

En ocasiones, cuando tenemos alguna colección de datos similares, requerimos aplicar un mismo procedimiento a todos estos datos. En tales casos se tiene que aplicar una función sobre todos los elementos de datos y se obtendrá como resultado un conjunto de elementos que contendrá a los datos modificados. Esto es, aplicando cierta función a un conjunto se obtiene como resultado un conjunto del mismo tamaño pero

que contiene el resultado de la aplicación de la función sobre cada uno de los elementos del conjunto original. Este concepto es similar al de la función *map* de Scheme y es una operación de conjuntos del tipo uno a uno. En R, la implementación de este concepto se logra mediante el uso de *funciones vectorizadas* y esta es la forma en la que se implementa el concepto de mapeo.

R contiene muchos operadores y funciones que están diseñados para manejar automáticamente vectores de números elemento a elemento sobre todo el objeto como si se tratara de una *repetición paralela.*

¿Cuál es la principal utilidad de las operaciones vectorizadas? Principalmente existen dos razones para utilizar las operaciones vectorizadas en R: *velocidad* y *claridad.* Un ciclo *for* puede alentar mucho la ejecución de un programa porque la expresión debe ser evaluada por el intérprete por cada iteración, por lo que es recomendable evitar usar ciclos cuando la ocasión lo permita y las funciones vectorizadas en R están codificadas en C, así que correrán mucho más rápido que un ciclo, además es más clara la sintaxis de una función vectorizada que todo el código que implica el cuerpo de un ciclo for. Por ejemplo, supongamos que a es un vector numérico, la expresión

```
log(a)
```

es más claro que la expresión:

```
for(i in 1:length(a)){
   a[i] <- log(a[i])
}
```

Finalmente, para poder aplicar el *mapeo* de cualquier función sobre un conjunto de datos R cuenta con la familia de comandos *apply* que producen el mismo resultado que un ciclo que itera sobre las entradas en un objeto, calculando la misma función sobre cada entrada, pero haciendo más claro el hecho de que se está haciendo lo mismo para cada entrada.

5.4. Interface con C y Fortran

Como sabemos, los lenguajes compilados proporcionan una mayor velocidad y eficiencia que un lenguaje interpretado como lo es R, cuya mayor virtud es su flexibilidad y robustez. Sin embargo, cuando se está trabajando con grandes cantidades de datos es imperativo conseguir esta rapidez y eficiencia en los cálculos.

Cuando un programa hace un uso intensivo de memoria, por ejemplo con recursión o ciclos anidados y no es posible la implementación con funciones vectorizadas, es recomendable mover esa porción de los cálculos a C (o Fortran) logrando así una

Clase R	Tipo de C	Tipo de Fortran
logical	int *	INTEGER
integer	int *	INTEGER
double	double *	DOUBLE PRECISION
complex	Rcomplex *	DOUBLE COMPLEX
character	char **	CHARACTER*255

Tabla 5.2: Correspondencias entre los objetos de R con los tipos en C y Fortran.

mejora en varios órdenes de magnitud. Si mantenemos el código del programa para Entrada/Salida y verificación de errores ahorraremos tiempo de programador.

Para usar las interfaces, es necesario conocer el mapeo entre los datos de los objetos de R y los tipos de argumentos en una función de C o una subrutina de Fortran. Los tipos de datos de C y de Fortran y sus correspondientes objetos de R están definidos en la siguiente tabla.

El código compilado para usarse en R es cargado como bibliotecas compartidas (.so en UNIX y .dll en Windows) lo cual le permite ser independiente de la plataforma.

El siguiente ejemplo muestra los conceptos de esta sección, es un ejemplo sencillo, pero servirá de ilustración.

```
#include "Rdefines.h"
#include <math.h> /* to declare extern double sin() */
void my_sin_vec(double *x,long *n) {
  long i ;
  for (i=0 ; i < *n ; i++)
    x[i] = sin(x[i]) ;
}
```

Esta función de C para calcular el seno toma un vector como argumento, por lo que es posible usarla para operaciones vectorizadas con lo que es seguro que de esta forma se pueden implementar funciones muy eficientes y rápidas.

Para llamar a una función de C, R cuenta con la función .C() y recibe el nombre de la función como cadena de caracteres y un argumento de R por cada argumento de C. Así, para poder utilizar la función *my_sin_vec* que se implementó en C, debemos pasarle el objeto (vector) sobre el cuál se harán los cálculos y su longitud de esta manera:

```
interface <- function(x,n)
{ .C("my_sin_vec", x = as.double(x),
    n = as.integer(length(x)))}
```

5.5. R y GUI

Una aplicación con una interfaz desde la línea de comandos (CLI, por sus siglas en ingles) es preferida por los usuarios experimentados en la aplicación porque permite un mejor control de las operaciones que se realizan, además de ser más flexible. Sin embargo, esto requiere un buen conocimiento de la sintaxis y restricciones de todas las funciones, lo que resulta intimidante para los principiantes. La curva de aprendizaje es mayor que con una interfaz gráfica (GUI), aunque el esfuerzo vale la pena y conduce a un mayor entendimiento del análisis, en este caso, además de que se pueden guardar las funciones y ejecutarlas en cualquier momento.

Una gran parte de los usuarios de genArise serían probablemente beneficiados con una interfaz gráfica de genArise, ya que en la mayoría de los casos se trata de investigadores que desean obtener resultados inmediatos para el análisis de algún microarreglo sobre el que están trabajando y no cuentan con el tiempo de investigar los parámetros de las funciones para la lectura de los datos de entrada, las funciones del preprocesamiento y el análisis o las funciones para escribir los datos en el disco. Pensando en esto, decidimos implementar una interfaz gráfica para genArise.

Para este propósito, R cuenta con extenciones que lo conectan con herramientas gráficas que pueden ser usadas para desarrollar menus y cajas de diálogo, como son: *Tcl/Tk*, Gtk, y wxPython. Se optó por realizar la implementación en Tcl/Tk, ya que después de analizar su conveniencia y desglosar las ventajas e inconvenientes de esta posibilidad se consideró que para el tamaño de la aplicación, las ventajas superaban a los inconvenientes. A continuación se presenta la lista:

Ventajas

- Sencillez de programación.
- Rapidez en el desarrollo de las aplicaciones.
- Multiplataforma.
- Gran velocidad comparado con otros lenguajes interpretados.
- Facilidad de modificación de las aplicaciones.
- Gran número de extensiones gratuitas (BWidgets, tkImage).

Inconvenientes

- Excesivamente lento comparado con los lenguajes compilados.
- Necesidad del intérprete para ejecutar una aplicación.
- Difícil de depurar, debido a que, a diferencia de un compilador, el intérprete sólo "traduce" el código que se ejecuta; pudiendo quedar partes del código sin depurar porque el intérprete nunca las haya ejecutado.

R ya contiene en el paquete base una extensión para Tcl/Tk, se trata de la biblioteca *tcltk* y es necesario contar con una instalación de *Tcl*. Sin embargo, es posible indicarle a R que se desea utilizar una instalación diferente de Tcl/Tk simplemente definiendo un par de variables de ambiente.

5.6. R y la programación orientada a objetos

La programación orientada a objetos es un estilo de programación muy popular actualmente. Gran parte de su popularidad se debe a que este paradigma hace más fácil escribir y mantener sistemas muy complejos. Los conceptos centrales de cualquier lenguaje de programación orientado a objetos son: la *clase* y los *métodos*. Una clase es la definición de un objeto. Típicamente una clase contiene varios atributos que son usados para almacenar información específica de esta clase. Un objeto es conocido también como una instancia de alguna clase.

Las operaciones son llevadas a cabo mediante los métodos, que son básicamente funciones que están especializadas en desarrollar un cálculo específico sobre los objetos de alguna clase en específico. Así es como funciona un lenguaje orientado a objetos. En R, las funciones genéricas son usadas para determinar el método apropiado. La función genérica es responsable de determinar la clase de sus argumentos y usa esa información para seleccionar el método apropiado.

Conclusiones

Una vez cumplidos los objetivos planteados al inicio de este trabajo, podemos concluir que genArise resulta una herramienta útil para realizar el análisis de los datos de microarreglos de ADN. Aunque ya existen varias herramientas con el mismo propósito, una de las ventajas más importantes del sistema genArise, reside en el hecho de que proporciona al usuario la opción de elegir entre los algoritmos más importantes y recomendados en la literatura para el preprocesamiento de los datos. Así también, cuenta con una interfaz gráfica que cumple los criterios ergonómicos permitiendo así un uso más intuitivo del sistema.

Otra ventaja es que genArise está implementado completamente en un lenguaje especializado en cálculos estadísticos, lo cual proporciona una mayor confiabilidad a los resultados pues este lenguaje cuenta ya con una gran cantidad de bibliotecas con procedimientos estadísticos.

Creemos importante mencionar que el hecho de que genArise haya sido liberado en un sitio internacional de gran importancia en el campo de la bioinformática, como Bioconductor, abre las puertas a que usuarios alrededor del mundo tengan la oportunidad de usar un sistema realizado en la UNAM. Esta fue nuestra principal motivación para compartir el software y continuar con su mantenimiento hasta ahora y que no resulte en un proyecto más de carácter local y sin seguimiento.

Al principio resultó complicado entender el significado biológico de los datos y los resultados que el sistema debía obtener; sin embargo, hemos visto que los resultados obtenidos con herramientas como esta proporcionan una información valiosa para diversos campos de investigación en busca de un beneficio para la población como lo es la medicina genómica surgida muy recientemente en el país. Durante el desarrollo del sistema nos dimos cuenta de que el trabajo multidisciplinario produce mejores resultados, ya que prermite la conjunción e interacción de las bases de diversas áreas en busca de un mismo fin.

genArise cuenta con características que lo situán como una nueva opción y que ha demostrado, después de una extensa fase de de pruebas y uso continuo por parte de algunos investigadores del Instituto de Fisiologia Celular, que obtiene resultados preliminares **up/down** tan confiables como otras herramientas en cuestión de minutos

y sin requerir tanto conocimiento previo del usuario. Estas características son:

- Es software libre, lo que significa que los usuarios tienen la libertad para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software sin pedir o pagar permisos.
- Para tener una mejor y más clara organización al realizar un análisis, genArise crea por cada uno de ellos, un proyecto que contiene por default dos directorios: uno llamado Plots y otro llamado Results, donde se almacenarán las gráficas y los resultados parciales, respectivamente.
- El formato de entrada de los datos que acepta genArise es muy flexible. Se trata de archivos donde cada columna representa un dato del microarreglo - por ejemplo, la señal Cy3 es una columna y su Background es otra columna, etc. - y cada una de ellas están separadas por tabuladores. Generalmente estos archivos de entrada se tratan de hojas de Excel.
- genArise conjunta diversos algoritmos para cada una de las operaciones del preprocesamiento, para permitir al usuario seleccionar el más apropiado para realizar una operación, de manera que los resultados del análisis se ajusten en mayor medida a las expectativas de su investigación.
- genArise brinda la oportunidad de realizar el preprocesamiento de los datos aplicando una secuencia de operaciones en el orden que comúnmente utiliza cualquier experimentalista, tan fácil como seguir el wizard; sin embargo, también existe la posibilidad de que sea el usuario el que determine este orden aplicando las operaciones una a una, ya que no existe una norma estándar en cuanto a la forma en que deben ser analizados estadísticamente los datos.
- Mediante la interfaz gráfica de genArise es posible visualizar diferentes gráficas de los datos tras cada una de las operaciones del preprocesamiento y el usuario puede elegir cuál de ellas desea desplegar. Además, estas gráficas se pueden guardar en el proyecto en formato pdf para su posterior consulta.
- genArise es una herramienta multiplataforma, lo que significa que corre en los sistemas operativos Windows y Linux y cuenta con una amplia documentación tanto sobre el funcionamiento como la instalación.

genArise se está convirtiendo en una herramienta de uso contínuo para el análisis de los datos obtenidos por la Unidad de Microarreglos del Institutot de Fisiología Celular, por ello uno de los planes a futuro es continuar con su mantenimiento, así como extenderlo, de tal manera que pueda ocupar un lugar al nivel de otras herramientas con el mismo propósito.

Bibliografía

[1] Regulación del Genoma Humano en México. Marcia Muñoz de Alba Medrano. Reflexiones en torno al derecho genómico. Universidad Nacional Autónoma de México.

[2] http://genome.gov]/Pages/EducationKit/download.html/ Glosario en Español

[3] La punta del iceberg. Enrique Reynaud. ¿Cómo ves?

[4] *Estrategias de análisis global. Hacia el manejo genético de las neoplasias.* Alfredo Hidalgo y Mauricio Salcedo. La Revista de Investigación Clínica, Vol. 53, Núm. 5. Septiembre-Octubre, 2001, pp 430-443.

[5] Smi th H O, Tomb J F, Dougherty B A et al. Frequency and distribution of DNA uptake signal sequences in the hemophilus influenzae Rd genome. Science. 1995. 269(5223):538-40

[6] Fraser C M, Gocayne J D, White O et al. The minimal gene complement of Mycoplasma genitalium. Science. 1995. 270(5235):397-403

[7] *Desarrollo de la Medicina genómica en México.* Gerardo Jiménez Sánchez, José Cuauhtémoc Valdés Olmedo, Guillermo Soberón. La Salud en Durango.

[8] (15) Adams M D, Celniker S E, Holt R A et al. The genome sequence of Drosophilla melangoster. Science. 2000. 287(5461):2185-95

[9] http://cifn.unam.mx/retlidb

[10] *Initial sequencing and analysis of the human genome* International Human Genome Sequencing Consortuim. Nature 2001; 409: pp 860-921.

[11] *The sequence of the human genome* Venter JC, *et al.* Science 2001; 291 pp 1255-7

[12] *La medicina genómica como un instrumento estratégico en el desarrollo de México* Dr. Gerardo Jiménez Sánchez. Ciencia y Desarrollo. Septiembre-Octubre de 2003. Volumen XXIX, Número 172 ISSN 0185-0008

[13] The lipid phosphatase SHIP2 controls insulin sensitivity. Clement S, Krause U, Desmedt F, Tanti JF, Behrends J, Pesesse X, Sasaki T, Penninger J, Doherty M, Malaisse W, Dumont JE, Le Marchand-Brustel Y, Erneux C, Hue L, Schurmans S. Nature. Enero 2001 4;409(6816):92-7.

[14] Genetic variation in the gene encoding calpain-10 is associated with type 2 diabetes mellitus. Horikawa et al. 2000. Nature Genetics 26, 163-175. 2000;26:163-175

[15] Nueva dimensión en el cuidado de la salud. El Instituto Nacional de Medicina Genómica, una alianza pública académica y privada. Gerardo Jiménez, José Cuauhtémoc Valdés, Guillermo Soberón. Caleidoscopio de la salud. Mexico 255-262 inmegen.org.mx

[16] Miroarreglos de DNA. Jorge Ramírez, Lorena chávez, José Luis Santillan y Simón Guzmán. Mensaje Bioquímico, Vol XXVII. Depto de Bioquímica, Facultad de Medicina, Universidad Nacional Autónoma de México, DF 2003.

[17] *La reacción en cadena de la polimerasa a dos décadas de su invensión.* Iram Pablo Rodríguez Sánchez, Hugo A. Barrera Saldaña.Ciencia UANL. Vol VII, No. 3. Julio-Septiembre 2004, pp 323-335.

[18] Microarrays y Biochips de ADN. Informe de Vigilancia Tecnológica. Genoma España. Salud Humana. *Fundación Española para el Desarrollo de la Investigación en Genómica y Proteómica/fundación General de la Universidad Autónoma de Madrid.* Octubre 2002

[19] Introducción a la regulación del inicio de la transcripción y circuitos genéticos. Rosa María Gutiérrez Ríos. Centro de Investigación sobre Fijación del Nitrógeno, UNAM. febrero 2004.

[20] Quackenbush, J. Microarray data normalization and transformation. Nature Genet. 32, 496-501 (2002).

[21] TIGR MIDAS Manual. Wei Liang, John Quackenbush. The Institute for Genomic Research. Noviembre 2004

[22] `http://www.r-project.org/about.html`

[23] *Robust locally weighted regression and smoothing scatterplots.* Cleveland, W. S. (1979), J. Amer. Statist. Assn. 74, 829-836.

[24] Engineering Statistics Handbook (en línea).4.1.4.4. LOESS (aka LOWESS) http://www.itl.nist.gov/div898/handbook/pmd/section1/pmd144.htm

[25] A. Aho, R. Sethi, J.D. Ullman. Compiladores: Principios, Técnicas y Herramientas, Addison-Wesley Iberoamericana, 1990.

[26] R Development Core Team. R Language Definition, 2002.

[27] Elisa Viso Gurovich, *Lenguajes de Programación I*, Fac. Ciencias, UNAM, 1987.

[28] J. Fox. Frames, Environments, and Scope in R and S-PLUS, Marzo 2002.

[29] R. Ihaka, R. Gentleman. R: A Language for Data Analysis and Graphics,1996.

[30] Bruce Eckel (2003), *Strong Typing vs. Strong Testing*, Thinking About Computing, http://www.mindview.net/WebLog/log-0025

Índice alfabético