RNAemo

Análisis de Requerimientos de Software

Franco Gaspar Riberi

10 de junio de 2012

Índice

1. Introducción

1.1. Propósito

El presente documento corresponde al análisis de requerimientos de la *iteración 1* de la tesis de grado referente a la carrera Licenciatura en Ciencias de la Computación de la UNRC, ***"RNAemo"***, a cargo de Franco Gaspar Riberi.

A continuación se detalla dicho análisis.

2. Problema

Actualmente no hay evidencia respecto a si existe diferencia en cuanto a reconocimiento de $_{mi}$RNA en los RNA$_m$ virales y sus supuestos humanizados. Por lo cual es necesario determinar la capacidad de hibridación de los microRNA en la secuencia viral tal como existe en la naturaleza y en la secuencia humanizada, para poder establecer conclusiones al respecto. En otras palabras, determinar si el uso de codones divergente con respecto al huésped podría ser una consecuencia de la presión evolutiva generada por los $_{mi}$RNA. Si eso es así, los $_{mi}$RNA deberían tener menor capacidad de unirse al RNA viral que al genoma viral humanizado.

En principio esa respuesta es importante desde el punto de vista biológico, y daría una herramienta importante para desarrollos posteriores, por ejemplo actualmente se está estudiando el uso de virus modificados para combatir cánceres, el programa podría predecir que virus sería menos afectado por los $_{mi}$RNA en células cancerosas y tenerlo en cuenta en el diseño del virus modificado.

2.1. Solución propuesta

Para determinar la divergencia en la hibridación de los $_{mi}$RNA en la secuencia viral original y humanizada, se contabilizará por separado la cantidad de $_{mi}$RNA que se hibridan a una secuencia viral y a la humanizada. Con esta información, se calcularán dos score de matching por cada genoma de las secuencias (viral y humanizada). El primero, dará un porcentaje de hibridación; el segundo, involucrará valores para las uniones A=T y G=C que permitirán calcular un score estimando aproximadamente la energía libre de estas uniones. Además se determinará si los nucleótidos del RNA$_m$ están apareados o no, determinando así que nucleótidos están disponibles para aparearse con el $_{mi}$RNA.

3. Algoritmos y Métodos

Para calcular el score de matching sobre las secuencias se utilizará la siguiente fórmula:

$$\frac{(\#AT \times constAT + \#GC \times constGC)}{(totalAT \times constAT + totalGC \times constGC} \quad (1)$$

donde:

- **#AT:** cantidad de Adenina que hace matching con Timina, o viceversa.
- **#GC:** cantidad de Guanina que hace matching con Citosina, o viceversa.
- **constAT:** valor predeterminado para el apareo A=T.
- **constGC:** análogo al anterior, pero con apareo G=C.
- **total AT:** total de adedina y timina (apareadas o no).
- **totalGC:** total de guanina y citosina (apareadas o no).

constAT y **constGC** tendrán diferentes valores según el score que se calcule. Si se calcula el score de matching en porcentaje, las constantes tendrán valor 1, pero en contrapartida, si se calcula el score de matchig teniendo en cuenta las uniones, se utilizarán los valores que se mencionan en la subsección 4.3.

4. Búsqueda de datos mínimos

Esta etapa consistió en la recolección de datos necesarios para posteriormente realizar las primeras pruebas del producto a desarrollar. Esta fase involucró la recolección de RNA mensajeros, $_{mi}$RNA, constantes para calcular el score de matching teniendo las uniones (A=T, G=C) y el software humanizador de secuencias.

4.1. RNA$_m$

La bases de datos biológicas son DNA céntricas, es decir que si un virus tiene su información como RNA, la secuencia archivada esta expresada como DNA (**T**imina en lugar de **U**racilo). Por lo cual, no se encontró RNA$_m$ directamente, sino que como gen.

Los virus obtenidos para las pruebas se obtuvieron de [3], el cual brinda el número de acceso a *GenBank*[1] para su obtención. Dentro de los diferentes códigos que se mencionan en [3] se obtuvieron los siguientes:

- Coxsackievirus A9 (Griggs; D00627).
- Enterovirus 71 (TW/2272/98; AF119795).
- Hepatitis A virus (HAF-203; AF268396).
- Poliovirus type 3 (23127; X04468).
- Rhinovirus type 89 (HRV89; M16248).
- Hepatitis E virus (Hyderabad; AF076239).
- Norwalk virus (BS5; AF093797).
- Astro-virus type 1 (Oxford; L23513).
- Dengue-1 virus (PDK-13; AF180818).
- Dengue-2 virus (S1; NC_001474).
- Dengue-3 virus (H87; M93130).
- Dengue-4 virus (NC_002640).
- GB virus C (G05BD; AB003292).
- Hepatitis C virus (HCV-A; AJ000009).
- Japanese encephalitis virus (GP78; AF075723).
- Murray Valley virus (MVE-1-51; AF161266).
- West Nile virus (2741; AF206518).
- Western tick-borne encephalitis virus (Neudoerfl; U27495).
- Yellow fever virus (Trinidad 79A; AF094612).

[1]Es una base de datos de secuencias genéticas. Una colección anotada de todas las secuencias de DNA disponibles al público. La base de datos GenBank está diseñada para proporcionar y fomentar el acceso de la comunidad científica a la mayor parte de información actualizada y completa la secuencia de DNA. Por lo tanto, *NCBI* (National Center for Biotechnology Information: http://www.ncbi.nlm.nih.gov/) no impone restricciones sobre el uso o la distribución de los datos GenBank.

- Eastern equine encephalitis virus (U01034).
- O'nyong-nyong virus (SG650; AF079456).
- Ross River virus (NB5092; M20162).
- Rubella virus (Cendehill; AF188704).
- Sindbis virus (Edsbyn 82-5; M69205).

A partir de los códigos de acceso, de la web: http://www.ncbi.nlm.nih.gov/genbank/ se obtuvieron los DNA de cada virus en formato FASTA. Para traducir los DNA a RNA (U por T), se utilizará la herramienta *BioEdit*. Alternativamente, se hará uso de la herramienta *Jalview*, o de la librería *Biopp*.

4.2. $_{mi}$RNA

Se obtuvo una base de datos de $_{mi}$RNA en formato FASTA. Dicha base de datos fue descargada de http://www.mirbase.org/cgi-bin/mirna_summary.pl?org=hsa. De la misma, se seleccionaron al azar aproximadamente 50 $_{mi}$RNA para luego realizar las primeras pruebas. Los $_{mi}$RNA están formados por 20 nucleótidos aproximadamente. A continuación se exhiben algunas de las secuencias:

- AAGAUGUGGAAAAAUUGGAAUC
- CAGUGGUUUUACCCUAUGGUAG
- CAGGCAGUGACUGUUCAGACGUC
- AUCAUAGAGGAAAAUCCAUGUU
- AAGAAGAGACUGAGUCAUCGAAU
- CUCUAGAGGGAAGCGCUUUCUG
- GUUUGCACGGGUGGGCCUUGUCU
- AAAGACCGUGACUACUUUUGCA
- UGUAAACAUCCCCGACUGGAAG
- UGGGCGAGGGGUGGGCUCUCAGAG

Otros posibles lugares de descarga de base de datos de $_{mi}$RNA consultados son:

- http://microrna.org/.
- http://mirdb.org/miRDB/.
- http://www.mirbase.org/.

4.3. Determinación de constantes para calcular score

Las constantes empleadas para el cálculos de score fueron extraídas de [7]. Las misma corresponden a valores razonables, dado que depende del contexto.

Los valores son:

Unión CG = ΔG(CG) = ΔG(GC) = -3
Unión AU = ΔG(AU) = ΔG(UA) = -2
Unión GU = ΔG(GU) = ΔG(UG) = -1

5. Control sobre las secuencias

Es necesario que las secuencias se encuentren en el marco de lectura correcto, de lo contrario significa que se perdió un nucleótido y todos los tripletes están cambiados. Para comprobar esto, es necesario traducir la cadena de nucleótidos a una cadena de aminoácidos. Esta tarea se realizará utilizado la herramienta *Bioedit*, como así también se hará uso de la librería *BioPP* la cual provee esta funcionalidad.

Al realizar el cambio, no deben aparecer codones stop (representados por un asterisco "*"). Si no aparecen codones stop significa que las secuencias están en el marco correcto de lectura (in frame).

6. Herramientas

A continuación se describen las herramientas y software necesarios para el desarrollo del producto.

6.1. R

R[2] es un entorno especialmente diseñado para el tratamiento de datos, cálculo y desarrollo gráfico. Permite trabajar con facilidad con vectores y matrices y ofrece diversas herramientas para el análisis de datos.

El lenguaje de programación R forma parte del proyecto GNU[3] y puede verse como una implementación alternativa del lenguaje S, desarrollado en AT&T Bell Laboratories. Se distribuye bajo la licencia GNU GPL y está disponible en una gran variedad de sistemas UNIX, como así también en Windows y MacOS.

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

[3] El proyecto GNU se inició en 1984 con el propósito de desarrollar un sistema operativo compatible con Unix que fuera de software libre. http://www.gnu.org/

R también es un entorno en el que se han ido implementando diversas técnicas estadísticas. Algunas de ellas se encuentran en la base de R pero muchas otras están disponibles como paquetes (packages[4]).

En resumen, R proporciona un entorno de trabajo especialmente preparado para el análisis estadístico de datos. Sus principales características son:

- Proporciona un lenguaje de programación propio. Basado en el lenguaje S, que a su vez tiene muchos elementos del lenguaje C.
- Objetos y funciones específicas para el tratamiento de datos.
- Es software libre. Permite la descarga de librerías con implementaciones concretas de funciones gráficas, métodos estadísticos, etcétera.

R será utilizado como software para graficar. Se utilizarán los scripts generados por SAARNA[5].

6.2. Software para Humanizar

GeneDesign[6] [5][6] será el software utilizado para humanizar las cadenas de nucleótidos. El mismo, corresponde a un software libre usado por biólogos moleculares, el gobierno y los sectores farmacéutico, químico, agrícola y biotecnológica para el diseño [4], el clon y validar las secuencias genéticas.

GeneDesign automatiza el proceso de determinar qué pares de bases deben ser unidos entre sí en un orden determinado para que un gen. Un gen codifica para una proteína específica, y el orden de los cientos o miles de pares de bases que componen ese gen determina el orden de los bloques de construcción de aminoácidos que componen esa proteína.

Consta de seis módulos que pueden ser utilizados individualmente o en serie para automatizar las tareas necesarias para diseñar y manipular secuencias de DNA sintético. El programa permite al usuario empezar con la secuencia del aminoácido que componen la proteína o las bases que forman el gen que codifica para esta proteína. Entonces el usuario se mueve a través de una serie de pasos que guían el diseño del gen y el vector que llevan el gen en la célula.

[4]Estos paquetes están disponibles en http://cran.au.r-project.org/.
[5]saarna.googlecode.com
[6]Descarga de: http://www.xmarks.com/site/slam.bs.jhmi.edu/gd/

6.3. BioEdit

BioEdit[7] es un editor de alineación de secuencias biológicas escrito para Windows 95/98/NT/2000/XP/7. Permite la alineación y manipulación de secuencias relativamente fácil. Puede utilizarse *wine* para su emulación en Linux.

6.4. Jalview

Jalview[8] es un editor de múltiples alineación escrito en Java, multiplataforma. El programa fue originalmente escrito por Michele Clamp. Se utiliza ampliamente por una variedad de servidores web (e.g., el servidor EBI ClustalW), pero está disponible como un editor de efectos de alineación general. *Jalview* tiene una amplia gama de funciones además de la alineación de secuencias múltiples, permite la visualización y edición incluyendo el cálculo de los árboles filogenéticos y la visualización de estructuras moleculares.

6.5. FASTA

FASTA [1] es un formato de archivo informático basado en texto, utilizado para representar secuencias de ácidos nucleicos, o de péptido, y en el que los pares de bases o los aminoácidos se representan usando códigos de una única letra. El formato también permite incluir nombres de secuencias y comentarios que preceden a las secuencias en sí.

Una secuencia bajo formato *FASTA* comienza con una descripción en una única línea (línea de cabecera), seguida por líneas de datos de secuencia. La línea de descripción se distingue de los datos de secuencia por un símbolo “>”. La palabra siguiente a este símbolo es el identificador de la secuencia, y el resto de la línea es la descripción (ambos son opcionales). No debería existir espacio entre el “>” y la primera letra del identificador. La secuencia termina si aparece otra línea comenzando con el símbolo “>”, lo cual indica el comienzo de otra secuencia.

A continuación se exhibe un ejemplo de una secuencia en tal formato:

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

[7] http://www.mbio.ncsu.edu/BioEdit/bioedit.html
[8] http://www.jalview.org/

6.5.1. Representación de secuencias

Cada línea de una secuencia debería tener algo menos de 80 caracteres. Las secuencias pueden corresponder a secuencias de proteínas o de ácidos nucleicos, y pueden contener huecos (o gaps) o caracteres de alineamiento.

- Los códigos de ácidos nucleicos soportados son:

Código ácido nucleico	**Significado**
-	hueco (gap) de longitud indeterminada
A	Adenosina
B	G T C (no A) (B viene tras la A)
C	Citosina
D	G A T (no C) (D viene tras la C)
G	Guanina
H	A C T (no G) (H viene tras la G)
K	G T (cetona/Ketone)
M	A C (grupo aMino)
N	A G C T (cualquiera/aNy)
R	G A (puRina)
S	G C (interacción fuerte/Strong interaction)
T	Timidina
U	Uracilo
V	G C A (no T, no U) (V viene tras la U)
W	A T (interacción débil/Weak interaction)
X	máscara
Y	T C (pirimidina/pYrimidine)

- Los códigos de aminoácidos soportados son:

Código aminoácido	Significado
A	Alanina
B	Ácido aspártico o Asparagina
C	Cisteína
D	Ácido aspártico
E	Ácido glutámico
F	Fenilalanina
G	Glicina
H	Histidina
I	Isoleucina
K	Lisina
L	Leucina
M	Metionina
N	Asparagina
O	Pirrolisina
P	Prolina
Q	Glutamina
R	Arginina
S	Serina
T	Treonina
U	Selenocisteína
V	Valina
W	Triptófano
Y	Tirosina
Z	Ácido glutámico o Glutamina
X	cualquiera
*	parada de traducción
-	hueco (gap) de longitud indeterminada

7. Bibliotecas existentes a utilizar

7.1. MiLi

MiLi[9] es una colección de pequeñas y útiles librerías desarrolladas en el lenguaje de programación C++ por **FuDePAN**, compuestas únicamente por cabeceras (*headers*). No requiere instalación para su uso y ofrece soluciones simples para problemas sencillos.

[9]MiLi: mili.googlecode.com.

Provee varias funcionalidades mediante archivos cabecera, conocidos en el ámbito de C/C++ como archivos con extensión "*.h*".

Dentro de las diversas funcionalidades podemos encontrar:

- **binary-streams:** permite serializar diferentes tipos de datos dentro de un único objeto utilizando los operadores de stream ($\ll$ y $\gg$). Hay dos maneras de utilizar esta librería:

  1. Empaquetar datos dentro de un objeto de salida (bostream) utilizando el operador $\ll$.
  2. Extraer datos desde un objeto de entrada (bistream) utilizando el operador $\gg$.

- **container-utils:** esta biblioteca provee un conjunto de funciones, optimizadas para cada tipo de contenedor STL:

  - find(container, element).
  - find(container, element, nothrow).
  - contains(container, element).
  - insert into(container, element).
  - copy container(from, to).

  Adicionalmente, esta biblioteca provee las siguientes clases:

  - ContainerAdapter<T>.
  - ContainerAdapterImpl<T, ContainerType>.

  Estos container adapters son una herramienta para lidiar con diferentes contenedores STL de manera homogénea, sin necesidad de conocer el tipo de contenedor que es (vector, list, map, set). Los usuarios pueden invocar al método INSERT (CONST T&) para insertar un elemento de tipo T sin la necesidad de saber el tipo del contenedor.

- **generic_exception:** ofrece una implementación de excepciones genéricas a partir de las cuales los desarrolladores pueden crear sus propias excepciones para problemas específicos de una manera muy simple.

7.2. FuD

FuD[10] es un framework para automatizar la implementación de aplicaciones distribuidas. Este framework es un sistema separado en las capas de *aplicación*, *administración* y *distribución* combinando los conceptos de cliente-servidor y Divide & Conquer. Tanto el cliente como el servidor se encuentran organizados en estos tres niveles separados, cada uno de ellos con una única responsabilidad bien definida. La comunicación entre los diferentes niveles se encuentra estrictamente limitada, es decir, por cada nivel existe un único punto de comunicación ya sea para comunicarse con la capa superior o con la inferior. Cuando un mensaje es creado, éste debe atravesar las diferentes capas comenzando desde la de nivel más alto hacia la capa inferior, y luego recorrer en sentido contrario las capas del lado receptor.

7.2.1. Diseño de original de FuD

- Application Layer (L3): proporciona los componentes que contienen todos los aspectos del dominio del problema a resolver. Estos aspectos incluyen todas las definiciones de los datos usados y su manipulación correspondiente, como así también todos los algoritmos relevantes para la solución al problema en general.

  Esta capa no es considerada como parte de FuD. La implementación de una aplicación que usa la librería, no es parte de la librería.

  Es necesario que del lado del servidor se implemente la aplicación principal, la cual hará uso de una simple interfaz en la abstracción de un trabajo distribuible permitiendo así codificar la estrategia de distribución de trabajos. Del lado cliente, solo se necesita implementar los métodos encargados de realizar las computaciones indicadas por una unidad de trabajo.

- Job Management Layer (L2): permite manejar los trabajos que se desean distribuir como así también generar las unidades de trabajo que serán entregadas a los clientes para su procesamiento. Estas unidades de trabajo llegan a su cliente correspondiente gracias a la capa más baja, encargada de la distribución. Una vez finalizado el procesamiento, se informa que todo ha terminado y otorga los resultados a la capa superior.

- Distributing Middleware Layer (L1): constituye un esquema de administración de clientes en particular. Tanto del lado cliente como

[10] **F**uDePAN **U**biquitous **D**istribution: fud.googlecode.com.

del servidor, la parte fija está dada por interfaces.

Las implementaciones concretas de este nivel son variables y están determinadas por el middleware a utilizar, por ejemplo BOOST.ASIO[11], MPI[12] o BOINC[13].

Actualmente, *FuD* cuenta con dos capas más, conocidas como *FuD-RecAbs* y *FuD-CombEng*.

7.3. BioPP

BioPP[14] es una biblioteca C++ para Biología Molecular. La misma proveerá las diversas estructuras de datos y métodos para manipular las secuencias de nucleótidos con las que se trabajará.

7.4. Biopp-filer

Biopp-filer[15] es una librería de persistencia para *Biopp*. Esta librería será utilizada para leer las secuencias en formato FASTA.

7.5. Odf-gen

Odf-gen[16] es una librería que ofrecerá funcionalidades que permiten generar archivos OpenDocument. Soporta diferentes tipos de archivos tales como:

- Hojas de cálculo. Compatible con OpenOffice Calc.
- Características compatibles: tablas, gráficos, gráficos de automóviles.
- APIs disponibles: C++, Python.

Será utilizada para generar un archivo en el cual se almacenarán las tablas comparativas. Alternativamente, las tablas respetarán el formato CSV.

[11] http://www.boost.org/doc/libs/1_4_00/doc/html/boost_asio.html
[12] http://www.mcs.anl.gov/research/projects/mpi/
[13] http://boinc.berkeley.edu/
[14] biopp.googlecode.com.
[15] biopp-filer.googlecode.com.
[16] odf-gen.googlecode.com.

7.6. Fideo

Fideo[17] es una librería que proveerá, entre otras, las funcionalidades necesarias para obtener la energía libre de una secuencia de nucleótidos. Casi la totalidad de su código proviene del proyecto VAC-O[18].

Esta librería será utilizada para la invocación externa de los programas de cálculo de estructura secundaria (folding e hibridación). Provee la abstracción necesaria para poder utilizar indistintamente cualquiera de ellos, ya sea mfold, unafold, vienna, entre otros.

8. Conclusiones

Como resultado de esta etapa de análisis se obtuvieron los datos mínimos necesarios para realizar las futuras pruebas del producto. Además se identificó la necesidad de extender la librería *fideo*. Por un lado, la inclusión de los backends mfold[8][9] y unafold[10], y por el otro, la inclusión de interfaces de hibridación de secuencias.

Asimismo, se observó que será importante almacenar el folding de los RNA_m, debido a que podrían utilizase como input en otra iteración.

Referencias

[1] *"FASTA"* http://en.wikipedia.org/wiki/FASTA_format

[2] *"A Framework for Small Distributed Projects and a Protein Clusterer Application"* Biset, Guillermo, 3 de diciembre de 2009.

[3] *"The extent of codon usage bias in human RNA viruses and its evolutionary origin"* Gareth M. Jenkins and Edward C. Holmes, 2003.

[4] *"Designing genes for successful protein expression."* Welch, et al., Methods Enzymol 2011 498:43-66.

[5] *"GeneDesign"* http://en.wikipedia.org/wiki/Gene_Designer

[6] *"GeneDesign"* http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1457031/?tool=pubmed

[7] *"Computational Methods for RNA Secondary Structure"* Zuker, Michael, June 8, 2006.

[17] fideo.googlecode.com.
[18] vac-o.googlecode.com

[8] *"The use of dynamic programming algorithms in RNA secondary structure prediction."* M. Zuker In M. S. Waterman, editor, Mathematical Methods for DNA Sequences. Boca Raton, FL, CRC Press, 1989.

[9] *"Mfold web server for nucleic acid folding and hybridization prediction."* M. Zuker, Nucleic Acids Res., 31(13):3406-3415, 2003.

[10] *"UNAFold: Software for Nucleic Acid Folding and Hybridization. In Data, Sequence Analysis, and Evolution"* N. R. Markham & M. Zuker., Bioinformatics: Volume 2, Humana Press Inc., 2008.