COLEGIO DE POSTGRADUADOS

BIOINFORMÁTICA

Aplicaciones a la genómica y proteómica

ISBN: 978-607-7533-81-8

Dr. Fernando Carlos Gómez Merino
Dra. Hilda Victoria Silva Rojas
Dr. Paulino Pérez Rodríguez

COORDINADORES

BIOINFORMÁTICA

Aplicaciones a la genómica y proteómica

Dr. Fernando Carlos Gómez Merino

Dra. Hilda Victoria Silva Rojas

Dr. Paulino Pérez Rodríguez

COORDINADORES

El Colegio de Postgraduados es un organismo público descentralizado del Gobierno Federal con personalidad jurídica y patrimonio propio, sectorizado en la Secretaría de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación (SAGARPA), cuyas actividades sustantivas son educación, investigación y vinculación encaminadas a la producción de alimentos nutritivos e inocuos, el manejo sustentable de los recursos naturales y al mejoramiento de la calidad de vida de la sociedad.

BIOINFORMÁTICA
Aplicaciones a la genómica y proteómica

Fernando Carlos Gómez Merino
Hilda Victoria Silva Rojas
Paulino Pérez Rodríguez

Carretera México-Texcoco km 36.5. Montecillo, 56230 Texcoco, Estado de México.

Miembro número 306 CANIEM

Año de publicación: 2010

ISBN: 978-607-7533-81-8
Edición electrónica

Hecho en México
Made in Mexico

Diseño de interiores y portada:
Paulino Pérez
Alfonso Nares Valle

www.colpos.mx

DIRECTORIO

COLEGIO DE POSTGRADUADOS

AGRADECIMIENTO A:

Línea Prioritaria de Investigación 5
Biotecnología Microbiana, Vegetal y Animal

Línea Prioritaria de Investigación 15
Estadística, Modelado y Tecnologías de Información Aplicadas a la Agricultura y al Medio Rural

COLEGIO DE POSTGRADUADOS

BIOINFORMÁTICA

Aplicaciones a la genómica y proteómica

Diego Riaño • Fidel Ramírez • Adrián Alexa • Elizabeth González • Flavia Vischi

Prólogo

La bioinformática es una disciplina que se origina a partir de los primeros análisis computacionales de secuencias de DNA y proteínas. Con el transcurrir del tiempo, el análisis de dichas secuencias sentó las bases de la bioinformática estructural, ahora en boga tras el éxito científico y tecnológico de diversos proyectos de secuenciación de genomas como el humano, de *Arabidopsis thaliana* y de arroz, y abordado principalmente por biólogos moleculares que desarrollan investigación en genómica y proteómica. Por otra parte, la bioinformática formal se enfoca al desarrollo de procesos informáticos para modelar y simular sistemas biológicos, así como al desarrollo y aplicación de algoritmos orientados al análisis de datos de distintas disciplinas científicas, para lo cual se aplican tanto métodos clásicos en bioinformática como de vida e inteligencia artificial. Además de técnicas de simulación, en este último enfoque se incluyen algoritmos bioinspirados o procedimientos computacionales inspirados en sistemas y fenómenos naturales como el crecimiento, el desarrollo, la reproducción, la evolución, la selección y la adaptación, entre otros.

La bioinformática es una ciencia multi e interdisciplinaria con sólidos fundamentos de las ciencias básicas (matemática, biología, física y química), de biología molecular, del funcionamiento básico de los dispositivos aplicados en informática y de computación aplicada a la biología de sistemas.

Gracias a esta ciencia es posible integrar tecnología, biología y computación para el desarrollo de programas útiles en alineamientos múltiples de secuencias, diseño de árboles filogenéticos, análisis tridimensional de moléculas, así como paquetes de programas para inferir filogenias por distintos métodos parsimonia, distancia matriz probabilidad; así como analizar secuencias de DNA y secuencias de proteínas, utilizando distintos tipos de algoritmos que permitan, entre otras opciones, comparar secuencias, predecir la estructura de genes, y acomodar posibles errores humanos o de secuenciación.

Gracias a los avances en la bioinformática ahora es más sencillo llevar a cabo análisis de genomas, transcriptomas, proteomas y metabolomas, desarrollar modelos computacionales para la visualización molecular, predicción de estructuras; simulación de metabolismo de virus por dinámica molecular, simulaciones estadísticas en sistemas biológicos que permitan la caracterización de problemas y obstáculos en situaciones reales, descubrimiento de nuevos medicamentos, aplicación de algoritmos de búsqueda de patrones exactos para la

detección de similitudes y aplicación de métodos computacionales y estadísticos en la caracterización poblacional, de relaciones filogenéticas y de evolución molecular. Con todo ello es posible modelar, previo análisis y comparación de genomas de especies silvestres, especies recombinantes que resulten más ventajosas, así como moléculas de interés alimenticio y farmacológico. Además, permite desarrollar estudios en metodologías estadísticas, matemáticas y computacionales para analizar genomas y expresión génica, así como pérdida de biodiversidad desarrollando modelos en los que se introduzcan variables para evaluar los posibles efectos del cambio climático global. Entre sus aplicaciones se encuentran el desarrollo e implementación de la tecnología de GeneChips, expresión génica, mapeo, rastreo de polimorfismos, descubrimiento de genes y desarrollo de algoritmos diagnósticos. Otras aplicaciones se concretan en simulaciones para neurociencia computacional, cristalografía macromolecular computacional, nutracéutica y nutragenómica, entre otras disciplinas.

En esta publicación intitulada "Bioinformática: aplicaciones a la genómica y proteómica", se detallan algunos de los avances más sobresalientes en los temas de genómica y proteómica, derivados de un curso internacional sobre el tema organizado por el Colegio de Postgraduados. Estos avances incluyen aspectos de las dos ciencias ómicas, incluyendo genómica y biología estructural, código R, análisis comparativo y evolución, agrupamiento y minería de datos en R, redes de interacciones entre proteínas y proteómica bioinformática.

Los coordinadores

Fernando C. Gómez Merino
Hilda Victoria Silva Rojas
Paulino Pérez Rodríguez

Índice general

1

Bases de bioinformática

Diego Mauricio Riaño Pachón

La bioinformática es una disciplina que surge de la interacción entre la biología, la estadística y las ciencias de la computación (Figura 1. Tiene como principales objetivos el manejo y análisis de grandes volúmenes de datos, principalmente producto de las nuevas tecnologías en biología molecular, como la genómica, la proteómica y la metabolómica, especialmente hoy en día con el advenimiento de nuevas tecnologías de secuenciación de ácidos nucleicos que están revolucionando la forma en como estudiamos los genomas. Otro aspecto importante incluye el desarrollo de nuevos métodos computacionales, algoritmos y/o software, para el análisis de esos datos.

Según Philip Bourne (UCSD), "la bioinformática se ha convertido en el interprete del lenguaje genómico del DNA y está intentando descifrar lenguajes más complejos en los que las proteínas son los sustantivos, las interacciones son la sintaxis, las rutas metabólicas son las oraciones y los sistemas vivos son el volumen completo" (Bourne, 2004).

Por lo tanto, de forma similar a la biología molecular, la bioinformática constituye hoy en día una caja de herramientas que todo investigador en biología tiene que manejar (Stein, 2008 presenta un punto de vista muy interesante).

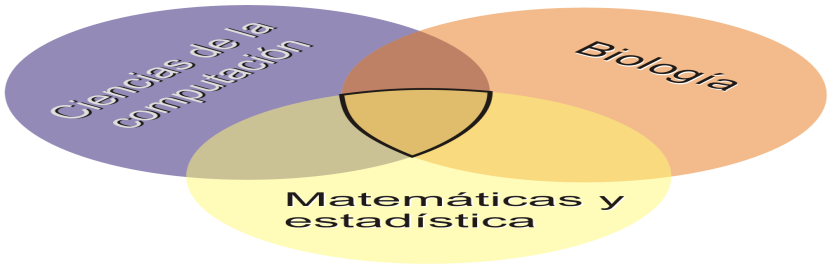


Fig. 1.1: La bioinformática es la disciplina que surge de la interacción de tres ciencias básicas: Biología, Matemáticas, y Ciencias de la computación. Cuando algunas de ellas dominan a las restantes se obtiene otra disciplina diferente de la bioinformática, por ejemplo, si matemática y biología son más importantes obtenemos biomatemáticas. Es importante que las tres ciencias base esten balanceadas para realizar proyectos de bioinformática.

En este curso nos concentraremos en el análisis de datos biológicos, usando, en la mayoría de los casos, herramientas de libre acceso, la mayoría de las cuales se desempeñan mejor en sistemas operativos tipo Unix[1].

[1]Linux, MacOSX, BSD, etc. Si quiere intentar tener una copia en su casa u oficina de alguno de esos sistemas operativos, recomiendo que use VirtualBox (u otra tecnología de virtualización), para instalar por ejemplo Linux dentro del sistema operativo existente, e.g., Windows XP; en un computador con por lo menos dos Cores y 2GB de RAM, de lo contrario es más conveniente tener un sistema Dual boot.

2

Herramientas bioinformáticas

2.1. Introducción al sistema Unix

El sistema operativo[1] es el conjunto de programas ("software") que sirve como interfaz entre la máquina ("hardware") y el usuario, y que le permite a este último ejecutar aplicaciones. Los sistemas operativos más comunes son: Windows (XP, Vista), Unix y MacOS X. Sistemas operativos tipo Unix (e.g., Linux) son usados principalmente en servidores, pero su uso en estaciones de trabajo y escritorios está en aumento. Las principales características de Unix son: multi-tarea, multi-usuario y portabilidad[2]. La mayoría de Unixes hoy en día tienen una interfaz gráfica amigable al usuario, desde la cual se pueden llevar a cabo casi todas las tareas de uso diario, como crear documentos, imprimir y navegar internet. Además de esta interfaz gráfica, existe una interfaz en línea de comandos que le permite al usuario ejecutar tareas mucho mas complejas y poderosas. A continuación vamos a aprender a usar la línea de comandos y algunos comandos que facilitan el manejo de archivos de gran tamaño, usando Linux como sistema operativo. Una guía sobre el uso de varios de esos comandos está disponible en el Apéndice [3].

2.1.1. La línea de comandos

A la línea de comandos se accede a través de un programa intérprete llamado "shell"[4]. Existen varios tipos de "shell" en Unix. en la mayoría de distribuciones Linux la "shell" bash viene instalada por defecto. Para usar la "shell" o línea de comandos de su computador, inicie el programa **Terminal**, que tiene un icono similar al que se muestra en la Figura 2.1.

Fig. 2.1: Icono del programa Terminal

[1] Más información en `http://en.wikipedia.org/wiki/Operating_system`

[2] Se refiere a que programas creados en diferentes Unixes pueden correr en uno u otro generalmente sin problema.

[3] Guías para otros programas usados comunmente en bioinformática están disponibles en `http://www.embnet.org/en/QuickGuides`

[4] `http://en.wikipedia.org/wiki/Unix_shell`

Al hacer click (o doble click, dependiendo de su configuración) en el icono iniciará el programa **Terminal**, similar al que se muestra en la Figura 2.2. Esta aplicación le da acceso a la línea de comandos de Linux a través de un *prompt*, que le indica que el sistema está esperando sus instrucciones. En la Figura 2.2, el *prompt* consiste de la cadena de caracteres `[user@server]$`, la cual consiste del nombre del usuario que está usando el programa **Terminal**, seguido del nombre de la máquina, y el símbolo dolar, inmediatamente después hay un cursor parpadeante a la espera de sus comandos.

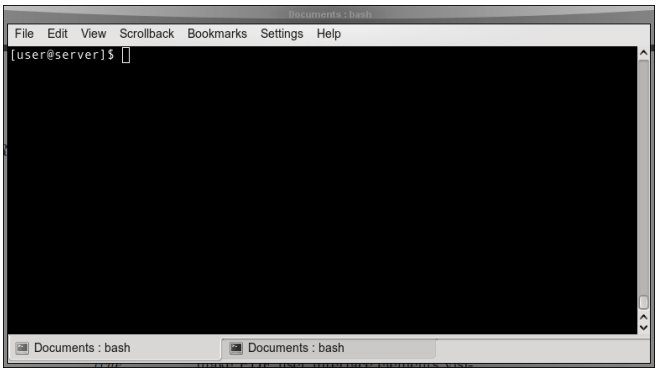


Fig. 2.2: Terminal en Linux

El *prompt* puede ser modificado alterando la variable de sistema `PS1`[5]. Vamos a cambiar el *prompt* para asegurarnos que todos tenemos el mismo.

En la sesión de **Terminal** ejecute los comandos como se muestran en el listado *Cambiando el prompt*. En la línea 4 salvamos el prompt en la nueva variable `SAVE`, en caso de que necesitemos recuperarlo. En línea 5 modificamos el *prompt* actual, `\u`[6], indica a nuestra "shell" mostrar el usuario actual, `\h`, muestra el nombre de la máquina y `\w`, muestra el directorio actual, el resto de caracteres se muestran sin ninguna modificación[7]. Compare su nuevo *prompt* (línea 6) con el antiguo (línea 1), el símbolo ~ hace referencia a su directorio casa, o directorio de usuario, en el sistema (vea Sección 2.1.2)

Cambiando el prompt

```
1 [user@server]$
2 [user@server]$ echo $PS1
3 [\u@\h]$
4 [user@server]$ SAVE=$PS1
5 [user@server]$ PS1="[\u@\h:\w]$ "
6 [user@server:~]$
```

Ahora que conoce su *prompt* y sabe como manipularlo, vamos a empezar a interactuar con el sistema a través de comandos. Para iniciar, ejecute el comando que se muestra en la línea 7, `wget` es un programa para descargar archivos de la red . Las líneas 8 a 17 muestran la salida

[5] `http://tldp.org/HOWTO/Bash-Prompt-HOWTO/c141.html`

[6] Lista de modificadores de *prompt* en bash: `http://tldp.org/HOWTO/Bash-Prompt-HOWTO/bash-prompt-escape-sequences.html`

[7] Ejercicio opcional: ¿Cómo hacer permanente el cambio de *prompt*?

típica de este comando, puede diferir ligeramente de la que se muestre en su **Terminal**. Cuando este comando termine, ejecute el que se muestra en la línea 19, que descomprime el archivo que acabó de descargar.

Descarga de archivos

```
7  [user@server:~]$ wget http://molbio00.bio.uni-potsdam.de/tmp/file1.tgz
8  --2009-07-27 12:56:25--  http://molbio00.bio.uni-potsdam.de/tmp/file1.tgz
9  Resolving molbio00.bio.uni-potsdam.de... 141.89.197.45
10 Connecting to molbio00.bio.uni-potsdam.de|141.89.197.45|:80... connected.
11 HTTP request sent, awaiting response... 200 OK
12 Length: 2413 (2.4K) [application/x-tar]
13 Saving to: `file1.tgz'
14
15 100%[========================== // ==>] 2,413       --.-K/s   in 0.006s
16
17 2009-07-27 12:56:25 (368 KB/s) - `file1.tgz' saved [2413/2413]
18
19 [user@server:~]$ tar xzf file1.tgz
```

2.1.2. Su casa y el árbol de directorios

Cada usuario en un sistema Unix tiene reservado un espacio, generalmente dentro del directorio "`/home`", en un subdirectorio que tiene el mismo nombre del usuario, e.g., para el usuario "diriano" su directorio personal es "`/home/diriano`", y recibe el nombre de directorio "casa", o directorio de usuario. La primera vez que inicia una sesión en Linux o en **Terminal**, se encuentra en su directorio casa. Si en algún momento no sabe en donde está, puede usar el comando que se muestra en la línea 20 para ubicar la ruta dentro del árbol de directorios en la que se encuentra. Es importante que note que los directorios utilizan el carácter "/" para referirse a una ruta de subdirectorios anidados, como se muestra en la línea 21 en el listado *Navegando el árbol de directorios*.

El árbol de directorios se refiere a la organización anidada de directorios en el sistema de archivos (Figura 2.3), similar a la organización de directorios en Microsoft Windows™que se puede visualizar con el **Explorador de Windows**.

Con el comando "listar" (Línea 22) muestra los directorios y archivos que se encuentran en el directorio actual. Este comando recibe argumentos/opciones que permiten obtener mas información sobre archivos y directorios. Una de las opciones más usadas es '`-l`' ("menos ele"; Línea24), cuya salida se muestra en las líneas 33 a 27, donde se muestra la lista de directorios en la ubicación actual, junto con los permisos sobre esos directorios, el número de subdirectorios, tamaño, fecha de ultima modificación y nombre.

Navegando al árbol de directorios

```
20 [user@server:~]$ pwd
21 /home/user
22 [user@server:~]$ ls
23 dia1 dia2
24 [user@server:~]$ ls -l
25 total 0
26 drwxr-xr-x  2 user  group  68 Aug  5 09:01 dia1/
27 drwxr-xr-x  2 user  group  68 Aug  5 09:02 dia2/
28 [user@server:~/dia1]$ cd dia1
```

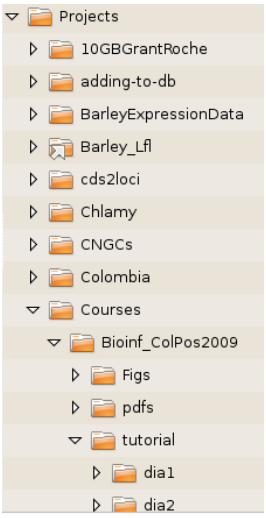


Fig. 2.3: Árbol de directorios en Linux

```
29 [user@server:~]$ cd ..
30 [user@server:~]$ cd /home/user/dia2/
```

Como se mencionó anteriormente, los sistemas Unix son multiusuario, lo que implica que debe existir un sistema de permisos en el sistema de archivos, para, por ejemplo, evitar pérdidas accidentales de datos, e.g., que un usuario elimine datos de otro. En la línea 27, se muestran los permisos del directorio `dia2` en la primera cadena de caracteres antes del primer espacio. El primer carácter indica si estamos ante n directorio (`d`), un archivo (`-`), o un enlace (`l`). Los siguientes 9 caracteres están divididos en 3 grupos de 3 caracteres cada uno, como se muestra en la Figura 2.4[8].

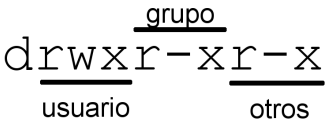


Fig. 2.4: Sistema de permisos en Linux. r: permiso de lectura; w: permiso de escritura; x: permiso de ejecución.

Ya sabemos como mostrar información sobre los directorios y archivos en la ubicación actual. Para cambiar de directorio usamos el comando '`cd nombre_directorio`', como se muestra en la línea 28. Si desea subir un nivel en la jerarquía de directorio, ejecute el comando `cd ..`, otra opción es usar la ruta absoluta del directorio al que se quiere llegar, como se muestra en la línea 30. Vuelva al subdirectorio `/home/usuario/dia1`.

Antes de continuar, quisiera presentar el comando más importante de cualquier sistema Unix, es el comando "manual", que muestra información sobre el uso de los diferentes comandos, por

[8] Ejercicio opcional: ¿Cómo cambiar los permisos de un archivo o directorio?

favor úsenlo cada vez que tengan alguna duda sobre las opciones o la sintaxis de algún comando, e.g., `man ls`.

2.1.3. Organizando archivos

Las operaciones más comunes con archivos son: copiar, mover y borrar. La sintaxis de los comandos para mover o copiar es la misma: "comando origen destino". Por ejemplo suponga que tiene un archivo llamado "test1.txt" en su directorio home y lo quiere mover al directorio "`~/dia1/`", tendría que ejecutar el comando que se muestra en la línea 37. Puede crear y remover directorios (vacíos) usando los comandos `mkdir` y `rmdir`, respectivamente.

Organizando archivos y directorios

```
31 [user@server:~]$ cd
32 [user@server:~]$ ls -l
33 total 0
34 drwxr-xr-x  2 user  group  68 Aug  5 09:01 dia1/
35 drwxr-xr-x  2 user  group  68 Aug  5 09:02 dia2/
36 -rw-r--r--  1 user  group   0 Aug 18 20:42 test1.txt
37 [user@server:~]$ mv test1.txt dia1/
38 [user@server:~]$ ls -l dia1/
39 total 0
40 -rw-r--r--  1 user  group  0 Aug 18 20:42 test1.txt
41 [user@server:~]$ ls -l
42 drwxr-xr-x  2 user  group  68 Aug  5 09:01 dia1/
43 drwxr-xr-x  2 user  group  68 Aug  5 09:02 dia2/
44 [user@server:~]$
```

2.1.4. Algunas operaciones básicas con archivos

Usando algunos comandos de UNIX podemos obtener información sobre archivos, y la información que ellos contienen, de forma rápida y eficiente, muchas veces no es necesario abrir el archivo, que puede ser de varios megabytes, para obtener esa información.

En el subdirectorio "`~/dia1/`", encuentra el archivo "TAIR9_pep_20090619", que corresponde a la base de datos de secuencias de proteínas predichas en el genoma de la planta modelo *Arabidopsis thaliana*. Para saber cuantas líneas tiene este archivo, ejecute el comando que se muestra en la línea 50.

¿A qué se deben las diferencias en las salidas de los comandos ejecutados en las líneas 50 y 52[9]?

Como se muestra en la línea 48, el tamaño de esta base de datos de secuencias es de 18′173,159 bytes. Para saber a cuanto corresponde esto en una unidad mas amigable use el comando que se muestra en la línea 54.

En la mayoría de ocasiones es importante ver como luce el archivo, ya sea al inicio o al final, pero debido al gran tamaño de los archivos con los que se trabaja normalmente, no es conveniente abrir el archivo con ningún editor de texto, ya que esto podría reducir el tiempo de respuesta del computador. Los comandos que se muestran en las líneas 58 y 69, muestran las 10 primeras y últimas líneas en el archivo, respectivamente.

[9] Revise la página de manual: `man wc`

Usando el comando `grep`, como se muestra en la línea 80, puede obtener un listado de las líneas en el archivo de interés que contienen un patrón dado, i.e., una cadena de texto específica.

Operaciones básicas con archivos

```
45 [user@server:~]$ cd dia1/
46 [user@server:~/dia1]$ ls -l
47 total 35496
48 -rw-r--r--  1 user group  18173159 Aug 30 16:14 TAIR9_pep_20090619
49 -rw-r--r--  1 user group         0 Aug 18 20:42 test1.txt
50 [user@server:~/dia1]$ wc TAIR9_pep_20090619
51   274243  790613 18173159 TAIR9_pep_20090619
52 [user@server:~]$ wc -l TAIR9_pep_20090619
53   274243 TAIR9_pep_20090619
54 [user@server:~/dia1]$ ls -lh
55 total 35496
56 -rw-r--r--  1 user group    17M Aug 30 16:14 TAIR9_pep_20090619
57 -rw-r--r--  1 user group     0B Aug 18 20:42 test1.txt
58 [user@server:~/dia1]$ head TAIR9_pep_20090619
59 >AT1G51370.2 | Symbols:  | F-box family protein
60 MVGGKKKTKICDKVSHEEDRISQLPEPLISEILFHLSTKDSVRTSALSTKWRYLWQSVPG
61 LDLDPYASSNTNTIVSFVESFFDSHRDSWIRKLRLDLGYHHDKYDLMSWIDAATTRRIQH
62 LDVHCFHDNKIPLSIYTCTTLVHLRLRWAVLTNPEFVSLPCLKIMHFENVSYPNETTLQK
63 LISGSPVLEELILFSTMYPKGNVLQLRSDTLKRLDINEFIDVVIYAPLLQCLRAKMYSTK
64 NFQIISSGFPAKLDIDFVNTGGRYQKKKVIEDILIDISRVRDLVISSNTWKEFFLYSKSR
65 PLLQFRYISHLNARFYISDLEMLPTLLESCPKLESLILVMSSFNPS*
66 >AT1G50920.1 | Symbols:  | GTP-binding protein-related
67 MVQYNFKRITVVPNGKEFVDIILSRTQRQTPTVVHKGYKINRLRQFYMRKVKYTQTNFHA
68 KLSAIIDEFPRLEQIHPFYGDLLHVLYNKDHYKLALGQVNTARNLISKISKDYVKLLKYG
69 [user@server:~/dia1]$ tail TAIR9_pep_20090619
70 LLRYLTI*
71 >ATMG00070.1 | Symbols: NAD9 | NADH dehydrogenase subunit 9
72 MDNQFIFKYSWETLPKKWVKKMERSEHGNRSDTNTDYLFQLLCFLKLHTYTRVQVSIDIC
73 GVDHPSRKRRFEVVYNLLSTRYNSRIRVQTSADEVTRISPVVSLFPSAGRWEREVWDMFG
74 VSFINHPDLRRISTDYGFEGHPLRKDLPLSGYVQVRYDDPEKRVVSEPIEMTQEFRYFDF
75 ASPWEQRSDG*
76 >ATMG00130.1 | Symbols: ORF121A | hypothetical protein
77 MASKIRKVTNQNMRINSSLSKSSTFSTRLRITDSYLSSPSVTELAPLTLTTGDDFTVTLS
78 VTPTMNSLESQVICPRAYDCKERIPPNQHIVSLELTYHPASIEPTATGSPETRDPDPSAY
79 A*
80 [user@server:~/dia1]$ grep ">" TAIR9_pep_20090619 | head -n 4
81 >AT1G51370.2 | Symbols:  | F-box family protein
82 >AT1G50920.1 | Symbols:  | GTP-binding protein-related
83 >AT1G36960.1 | Symbols:  | unknown protein
84 >AT1G44020.1 | Symbols:  | DC1 domain-containing protein
```

No siempre en bioinformática tratamos con secuencias, en muchas ocasiones tenemos datos en forma tabular, donde los campos están separados por algún carácter definido, e.g., tabulación o comas. En la mayoría de los casos esto implica almacenar y manejar los datos usando un sistema de bases de datos, como MySQL. Sin embargo, es importante hacerse una idea de los resultados antes de integrarlos en el sistema de bases de datos, una opción que ha aparecido recientemente, dirigida a biólogos que trabajan con grandes cantidades de datos, es el Scriptome[10], en el cual el autor ofrece una colección de "scripts" de PERL que se pueden ejecutar en la línea de comandos. En la líneas 85 a 87 se ve un ejemplo en que se cambian todos los

[10] http://sysbio.harvard.edu/csb/resources/computational/scriptome/UNIX/

caracteres a mayúsculas, el comando se tiene que ejecutar en una única línea, aquí se muestra en líneas separadas solo para facilitar su visualización.

Ejemplo de Scriptome

```
85 [user@server:~/dia1]$ perl -e ' while(<>) {print lc($_);} \
86 warn "Changed $. lines to lower case\n" ' \
87 TAIR9_pep_20090619 > TAIR9_pep_20090619.lc
88 changed 274243 lines to lower case
89 [user@server:~/dia1]$ ls -l
90 total 70992
91 -rw-r--r--  1 user  group  18173159 Aug 30 16:14 TAIR9_pep_20090619
92 -rw-r--r--  1 user  group  18173159 Aug 30 19:54 TAIR9_pep_20090619.lc
93 -rw-r--r--  1 user  group         0 Aug 18 20:42 test1.txt
94 [user@server:~/dia1]$ head -n 2 TAIR9_pep_20090619.lc
95 >at1g51370.2 | symbols:  | f-box family protein
96 mvggkkktkicdkvsheedrisqlpepliseilfhlstkdsvrtsalstkwrylwqsvpg
97 [user@server:~/dia1]$
```

En la línea 80 usó el símbolo "|" o "barra vertical", o "tubería" en UNIX, le permite conectar comandos, de forma que la salida del formato a la izquierda de la barra vertical sirve de entrada al comando que está a la derecha de la barra. Y en la línea 87 usó el símbolo ">" para redireccionar la salida estándar del comando hacia un archivo.

2.2. Formatos de secuencias

Existen diferentes formatos para secuencias, generalmente en texto plano. Lo que significa que se pueden ver y editar con cualquier editor de texto, como `vi` o `pico`. Algunos de estos formatos son más comunes que otros y muchos programas de bioinformática aceptan varios de los formatos más comunes (Leonard *et al.*, 2007).

Todos los formatos de secuencias tienen una característica (campo) en común: un identificador para cada secuencia. De forma que esta pueda ser reconocida de forma unívoca.

2.2.1. Fasta

El formato más sencillo es conocido como Fasta[11]. En el cual una entrada y una secuencia, se pueden dividir en dos partes: La línea de identificación, que **debe** comenzar con el símbolo ">" y seguida inmediatamente del identificador de la secuencia (Ver línea 98), que puede ser cualquier cadena de caracteres sin espacios. Las líneas inmediatamente después del identificador corresponden a la secuencia propiamente dicha (Líneas 99-105).

Fasta es el formato de secuencias más comúnmente usado en aplicaciones bioinformáticas.

Secuencia en formato FastA

```
98  >gi|110742030|dbj|BAE98952.1| putative NAC domain protein [Arabidopsis thaliana]
99  MEDQVGFGFRPNDEELVGHYLRNKIEGNTSRDVEVAISEVNICSYDPWNLRFQSKYKSRDAMWYFFSRRE
100 NNKGNRQSRTTVSGKWKLTGESVEVKDQWGFCSEGFRGKIGHKRVLAFLDGRYPDKTKSDWVIHEFHYDL
101 LPEHQRTYVICRLEYKGDDADILSAYAIDPTPAFVPNMTSSAGSVVNQSRQRNSGSYNTYSEYDSANHGQ
102 QFNENSNIMQQQPLQGSFNPLLEYDFANHGGQWLSDYIDLQQQVPYLAPYENESEMIWKHVIEENFEFLV
103 DERTSMQQHYSDHRPKKPVSGVLPDDSSDTETGSMIFEDTSSSTDSVGSSDEPGHTRIDDIPSLNIIEPL
```

[11] http://www.ncbi.nlm.nih.gov/blast/fasta.shtml

```
104 HNYKAQEQPKQQSKEKVISSQKSECEWKMAEDSIKIPPSTNTVKQSWIVLENAQWNYLKNMIIGVLLFIS
105 VISWIILVG
```

2.2.2. GenBank

El formato GenBank[12][13] es usado por el "National Center for Biotechnology Information" (NCBI[14]), el mayor repositorio de secuencias, tanto de ácidos nucleicos como de proteínas, a nivel mundial. El NCBI junto con el, EMBL[15] y el DDBJ[16], mantienen en forma conjunta "The International Nucleotide Sequence Database" (Mizrachi, 2008).

Una entrada en este formato está compuesta por dos partes. La primera parte consiste de las posiciones 1 a 10, y generalmente contiene el nombre del campo, e.g., `LOCUS`, `DEFINITION`, `ACCESSION` o `SOURCE`. La segunda parte de cada entrada contiene la información para el campo correspondiente. Cada entrada termina con el símbolo "\\" (Línea 168).

Secuencia en formato GenBank

```
106 LOCUS       BAE98952                429 aa            linear   PLN 27-JUL-2006
107 DEFINITION  putative NAC domain protein [Arabidopsis thaliana].
108 ACCESSION   BAE98952
109 VERSION     BAE98952.1  GI:110742030
110 DBSOURCE    accession AK226863.1
111 KEYWORDS    .
112 SOURCE      Arabidopsis thaliana (thale cress)
113   ORGANISM  Arabidopsis thaliana
114             Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
115             Spermatophyta; Magnoliophyta; eudicotyledons; core eudicotyledons;
116             rosids; eurosids II; Brassicales; Brassicaceae; Arabidopsis.
117 REFERENCE   1
118   AUTHORS   Totoki,Y., Seki,M., Ishida,J., Nakajima,M., Enju,A., Morosawa,T.,
119             Kamiya,A., Narusaka,M., Shin-i,T., Nakagawa,M., Sakamoto,N.,
120             Oishi,K., Kohara,Y., Kobayashi,M., Toyoda,A., Sakaki,Y.,
121             Sakurai,T., Iida,K., Akiyama,K., Satou,M., Toyoda,T., Konagaya,A.,
122             Carninci,P., Kawai,J., Hayashizaki,Y. and Shinozaki,K.
123   TITLE     Large-scale analysis of RIKEN Arabidopsis full-length (RAFL) cDNAs
124   JOURNAL   Unpublished
125 REFERENCE   2  (residues 1 to 429)
126   AUTHORS   Totoki,Y., Seki,M., Ishida,J., Nakajima,M., Enju,A., Morosawa,T.,
127             Kamiya,A., Narusaka,M., Shin-i,T., Nakagawa,M., Sakamoto,N.,
128             Oishi,K., Kohara,Y., Kobayashi,M., Toyoda,A., Sakaki,Y.,
129             Sakurai,T., Iida,K., Akiyama,K., Satou,M., Toyoda,T., Konagaya,A.,
130             Carninci,P., Kawai,J., Hayashizaki,Y. and Shinozaki,K.
131   TITLE     Direct Submission
132   JOURNAL   Submitted (26-JUL-2006) Motoaki Seki, RIKEN Plant Science Center;
133             1-7-22 Suehiro-cho, Tsurumi-ku, Yokohama, Kanagawa 230-0045, Japan
134             (E-mail:mseki@psc.riken.jp, URL:http://rarge.gsc.riken.jp/,
135             Tel:81-45-503-9625, Fax:81-45-503-9586)
136 COMMENT     An Arabidopsis full-length cDNA library was constructed essentially
137             as reported previously (Seki et al. (1998) Plant J. 15:707-720;
138             Seki et al. (2002) Science 296:141-145).
139             This clone is in a modified pBluescript vector.
140             Please visit our web site (http://rarge.gsc.riken.jp/) for further
141             details.
142 FEATURES             Location/Qualifiers
143      source          1..429
144                      /organism="Arabidopsis thaliana"
145                      /db_xref="taxon:3702"
146                      /chromosome="1"
147                      /clone="RAFL08-19-M04"
148                      /ecotype="Columbia"
149                      /note="common name: thale cress"
150      Protein         1..429
151                      /product="putative NAC domain protein"
```

[12] http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html
[13] ftp://ftp.ncbi.nih.gov/genbank/release.notes/gb172.release.notes
[14] http://www.ncbi.nlm.nih.gov/
[15] http://www.ebi.ac.uk/embl/
[16] http://www.ddbj.nig.ac.jp/

```
152       Region          5..137
153                       /region_name="NAM"
154                       /note="No apical meristem (NAM) protein; pfam02365"
155                       /db_xref="CDD:111274"
156       CDS             1..429
157                       /gene="At1g01010"
158                       /coded_by="AK226863.1:89..1378"
159   ORIGIN
160           1 medqvgfgfr pndeelvghy lrnkiegnts rdvevaisev nicsydpwnl rfqskyksrd
161          61 amwyffsrre nnkgnrqsrt tvsgkwkltg esvevkdqwg fcsegfrgki ghkrvlafld
162         121 grypdktksd wvihefhydl lpehqrtyvi crleykgdda dilsayaidp tpafvpnmts
163         181 sagsvvnqsr qrnsgsynty seydsanhgq qfnensnimq qqplqgsfnp lleydfanhg
164         241 gqwlsdyidl qqqvpylapy enesemiwkh vieenfeflv dertsmqqhy sdhrpkkpvs
165         301 gvlpddssdt etgsmifedt ssstdsvgss depghtridd ipslniiepl hnykaqeqpk
166         361 qqskekviss qksecewkma edsikippst ntvkqswivl enaqwnylkn miigvllfis
167         421 viswiilvg
168   //
```

Algunas operaciones básicas con secuencias en formato Fasta

En lo que resta de esta sección, y la siguiente, solo usaremos secuencias en formato Fasta. Por favor verifique que las secuencias de *A. thaliana* en el archivo `TAIR9_pep_20090619` están en este formato. Puede usar el comando "`head nombre_archivo`", o el comando "`less nombre_archivo`"[17].

¿Algunas vez ha tenido que contar el número de secuencias o cambiar el identificador de secuencias en formato Fasta? Si se trata de una docena de secuencias, esto se podría hacer fácilmente en cualquier editor de textos, pero cuando son miles de secuencias la opción del editor de textos dejar de ser viable. Afortunadamente algunos comandos de Unix nos permiten realizar estas tareas simples de forma rápida.

Como observó en la línea 80, el comando "`grep`" nos podría ser de ayuda para contar el número de secuencias en un archivo Fasta. el modificador "`-c`" cuenta el número de líneas que contienen un patrón dado en un archivo, y podemos aprovechar el hecho de que en un archivo Fasta el símbolo "`>`" aparece una única vez por cada secuencia, como se muestra en la línea 175.

Usando comandos Unix con archivos Fasta

```
169   [user@server:~]$ cd ~/dia1/
170   [user@server:~/dia1]$ ls -l
171   total 70992
172   -rw-r--r--  1 user  group  18173159 Aug 30 16:14 TAIR9_pep_20090619
173   -rw-r--r--  1 user  group  18173159 Aug 30 19:54 TAIR9_pep_20090619.lc
174   -rw-r--r--  1 user  group         0 Aug 18 20:42 test1.txt
175   [user@server:~/dia1]$ grep -c ">" TAIR9_pep_20090619
176   33410
177   [user@server:~/dia1]$ sed 's/>/>ATH_/' TAIR9_pep_20090619 > TAIR9_pep_20090619.mod
178   [user@server:~/dia1]$ head TAIR9_pep_20090619.mod
179   >ATH_AT1G51370.2 | Symbols:
180   MVGGKKKTKICDKVSHEEDRISQLPEPLISEILFHLSTKDSVRTSALSTKWRYLWQSVPG
181   LDLDPYASSNTNTIVSFVESFFDSHRDSWIRKLRLDLGYHHDKYDLMSWIDAATTRRIQH
182   [user@server:~/dia1]$
```

En otras ocasiones es importante modificar el identificador de cada secuencia, de forma que este incluya, por ejemplo, una abreviatura que represente el nombre de la especie a la que pertenece la secuencia. Nuevamente Unix nos permite hacer este cambio muy rápido usando el comando `sed` como se muestra en la línea 177.

[17] Para salir de `less` presione "`q`"

2.3. Aplicaciones bioinformáticas

Ya que sabemos como utilizar un sistema Unix y varios de sus comandos para desarrollar tareas simples con archivos de secuencias, ahora podemos comenzar con las aplicaciones que se han desarrollado específicamente para bioinformática.

2.3.1. EMBOSS

EMBOSS[18], "The European Molecular Biology Open Software Suite", en un paquete gratuito, con código fuente abierto, compuesto de cientos de aplicaciones[19] que se han desarrollado específicamente para resolver las necesidades de la comunidad de biología molecular. El tutorial que se encuentra a continuación es una parte de los tutoriales disponibles en `http://emboss.sourceforge.net/docs/emboss_tutorial/emboss_tutorial.html`.

Algunos de las áreas cubiertas por aplicaciones de EMBOSS son:

- Alineamiento de secuencias
- Búsqueda en bases de datos usando patrones
- Identificación de motivos de proteínas
- Análisis de uso de codones.

El programa "`wossname`" nos permite encontrar otros programas en EMBOSS, mediante la búsqueda de palabras claves en la descripción de esos programas. Si se omiten las palabras clave, `wossname` presenta una lista de todos los programas en EMBOSS. Muchos de los programas en EMBOSS tienen opciones adicionales que se pueden controlar mediante modificadores en la línea de comandos, para obtener información sobre esas opciones use, por ejemplo, "`tfm wossname`", el cual lo lleva a la página de manual del programa. Uno de los modificadores más comunes y mas útil para nuevos usuarios es "`-opt`", que hace que el programa muestre de forma interactiva todas sus opciones, intente con "`wossname -opt`"

Usando EMBOSS

```
[user@server:~]$ cd ~/dia1/
[user@server:~/dia1]$ wossname alignment
Finds programs by keywords in their short description
SEARCH FOR 'ALIGNMENT'
aligncopy     Reads and writes alignments
aligncopypair Reads and writes pairs from alignments
cons          Create a consensus sequence from a multiple alignment
consambig     Create an ambiguous consensus sequence from a multiple alignment
diffseq       Compare and report features of two similar sequences
...
```

El programa "`seqret`" nos permite extraer secuencias de una base de datos, y convertir de un formato de secuencias a otro. En el directorio "`~/dia1/`", encuentra la secuencia

[18] `http://emboss.sourceforge.net/`
[19] `http://emboss.sourceforge.net/apps/release/6.1/emboss/apps/`

"BAE98952.pep.gb", con la proteína de un factor de transcripción en formato GenBank, use el comando que se muestra en la línea 197 para convertir a formato fasta. En caso de que solo necesite extraer una secuencia (o varias) de un archivo, puede usar "seqret" como se muestra en la línea 203.

Comandos básicos de EMBOSS

```
193 [user@server:~]$ cd ~/dia1/
194 [user@server:~/dia1]$ ls -l BAE98952.pep.gb
195 -rw-r--r--  1 user  group  3426 Sep  6 19:22 BAE98952.pep.gb
196 [user@server:~/dia1]$ less BAE98952.pep.gb
197 [user@server:~/dia1]$ seqret BAE98952.pep.gb fasta::BAE98952.pep.fa
198 Reads and writes (returns) sequences
199 [user@server:~/dia1]$ head -n 3 BAE98952.pep.fa
200 >BAE98952 BAE98952.1 putative NAC domain protein [Arabidopsis thaliana].
201 medqvgfgfrpndeelvghylrnkiegntsrdvevaisevnicsydpwnlrfqskyksrd
202 amwyffsrrennkgnrqsrttvsgkwkltgesvevkdqwgfcsegfrgkighkrvlafld
203 [user@server:~/dia1]$ seqret TAIR9_pep_20090619.mod:ATH_AT1G01010.1
204 Reads and writes (returns) sequences
205 output sequence(s) [ath_at1g01010.fasta]:
206 [user@server:~/dia1]$ infoseq BAE98952.pep.gb
207 Display basic information about sequences
208 USA                     Database  Name           Accession      Type Length Description
209 genbank::BAE98952.pep.gb:BAE98952 -              BAE98952       BAE98952       P    429    putative NAC domain protein
210 [user@server:~/dia1]$ infoseq TAIR9_pep_20090619.mod -only -length -auto > TAIR9_pep_20090619.length
```

El programa seqret permite tener como entrada un archivo con listas de referencias a bases de datos e identificadores (Línea 211), el que se usa agregando el símbolo @ antes del nombre del archivo de lista, como se muestra en la línea 218.

El programa "infoseq", proporciona información sobre la secuencia o secuencias (Línea 206). Puede usarlo para extraer los datos necesarios para crear un histograma de la distribución de la longitud de secuencias de proteínas en el genoma de *A. thaliana* usando R[20] (Línea 210).

El programa "compseq", le permite calcular la composición de palabras de longitud dada en una secuencia. Por ejemplo, puede calcular las frecuencias de monómeros o dímeros en todos los transcritos anotados en el genoma de *A. thaliana*, como se muestra en la línea 221, revise la salida de este programa usando less. ¿Cómo se comportan las frecuencias de monómeros en los cDNA de *A. thaliana*?

Usando el programa "seqret" extraiga la secuencia ATH_AT4G01540.1 (Línea 226), esta secuencia corresponde a otro factor de transcripción de la familia NAC, que está usualmente anclado a sistemas de membranas[21]. Vamos a usar el programa "tmap", para predecir la región de la proteína que se ancla a la membrana (Línea 229), parte de la salida de tmap es gráfica como se muestra en la Figura 2.5. En la figura vemos que hay una región transmembranal en el extremo C-terminal.

Análisis de secuencias con EMBOSS

```
211 [user@server:~/dia1]$ cat @ATH_sigma70like.lista.txt
212 TAIR9_pep_20090619.mod:ATH_AT1G08540.1
213 TAIR9_pep_20090619.mod:ATH_AT1G64860.1
```

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

[21] http://www.plantcell.org/cgi/content/full/18/11/3132

```
214 TAIR9_pep_20090619.mod:ATH_AT2G36990.1
215 TAIR9_pep_20090619.mod:ATH_AT3G53920.1
216 TAIR9_pep_20090619.mod:ATH_AT5G13730.1
217 TAIR9_pep_20090619.mod:ATH_AT5G24120.1
218 [user@server:~/dia1]$ seqret @ATH_sigma70like.lista.txt -outseq ATH_sigma70like.fa
219 Reads and writes (returns) sequences
220 [user@server:~/dia1]$ less ATH_sigma70like.fa
221 [user@server:~/dia1]$ compseq TAIR9_pep_20090619.mod
222 Calculate the composition of unique words in sequences
223 Word size to consider (e.g. 2=dimer) [2]: 1
224 Program compseq output file [at1g51370.composition]:
225 [user@server:~/dia1]$ less at1g51370.composition
226 [user@server:~/dia1]$ seqret TAIR9_pep_20090619.mod:ATH_AT4G01540.*
227 Reads and writes (returns) sequences
228 output sequence(s) [ath_at4g01540.fasta]:
229 [user@server:~/dia1]$ tmap ath_at4g01540.fasta
230 Predict and plot transmembrane segments in protein sequences
231 Graph type [x11]:
232 Output report [ath_at4g01540.tmap]:
233 [user@server:~/dia1]$ pepinfo ath_at4g01540.fasta
234 Plot amino acid properties of a protein sequence in parallel.
235 Graph type [x11]:
236 Output file [ath_at4g01540.pepinfo]:
237 [user@server:~/dia1]$
```

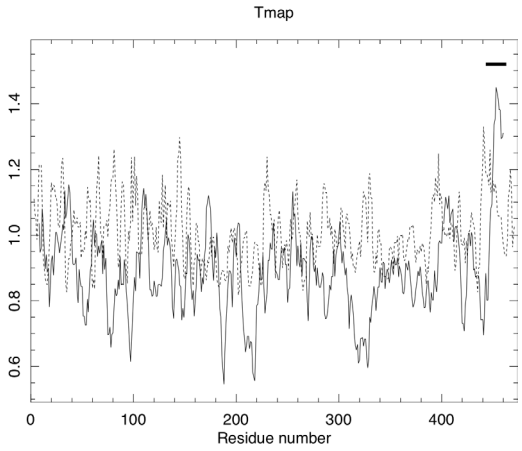


Fig. 2.5: Análisis de regiones transmembranales de AT4G01540.1. La línea continua representa la tendencia de la secuencia a ser parte de una región transmembranal interna. La línea punteada representa la tendencia de pertenecer a la parte final de una región transmembranal.

Con el programa "`pepinfo`" (Línea 233) podemos ver los patrones de hidrofobicidad a lo largo de la secuencia, como se muestra en la Figura 2.6, en la que vemos que el mayor pico de hidrofobicidad se encuentra también en el extremo C-terminal de la proteína.

El programa Primer3 es uno de los más usados para el diseño de iniciadores en PCR. EMBOSS tienen una interface para usar primer3, es el programa "`eprimer3`", usando con la opción `-opt` puede modificar varias de las opciones con las que viene por defecto. Vea la página de

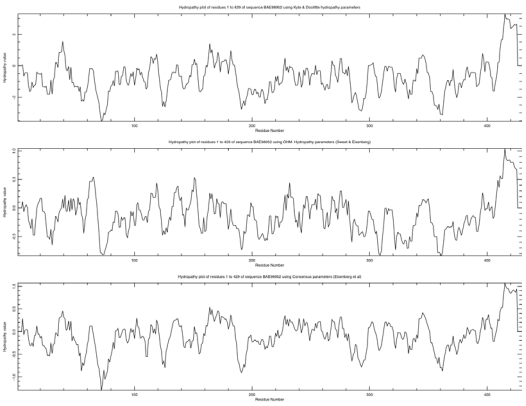


Fig. 2.6: Análisis de hidrofobicidad de AT4G01540.1

manual si necesita modificar otras opciones. Usando `eprimer3` puede diseñar iniciadores para cientos de secuencias, y puede integrarlo en una línea de procesamiento de datos que haga verificaciones de los cebadores propuestos. Nosotros hemos implementado esta idea en la herramienta gráfica QUANTPRIME[22].

Vamos a terminar esta corta introducción a EMBOSS trabajando con los programas "`fuzznuc`" y "`fuzzpro`". La función de ambos es buscar patrones en secuencias, el primero en ácidos nucleicos y el segundo en proteínas. Los patrones de búsqueda se definen usando la sintaxis de PROSITE[23]. Por ejemplo, el patrón `[FY]-[LIV]-G-[DE]-E-A-Q-x-[RKQ](2)-G` se interpreta de la siguiente forma: la primera posición puede contener o una Fenilalanina (F) a o una Tirosina (Y), el segundo sitio puede contener L o I o V, y así sucesivamente.

Busque el motivo "`R-R-[ILV]-Y-D-[IAV]-[TVAL]-N-[VI]-[LF]`" usando `fuzzpro`, en el archivo que contiene todas las proteínas anotadas en el genoma de *A. thaliana* (Línea 238-240), use `-rformat2` para cambiar el formato del reporte de salida[24].

La Figura 2.7 representa dos elementos reguladores de la transcripción en promotores de genes vegetales (GOMEZ-PORRAS *et al.*, 2007). Vamos a usar la matriz que representa el motivo ABRE para crear un patrón de búsqueda. Para cada una de las posiciones (columnas) tome la base mas frecuente y construya un patrón similar al que usó anteriormente, use el programa "`fuzznuc`" (Línea 243-245) para buscar este patron en la base de datos de promotores[25] de Arabidopsis (`TAIR9_upstream_1000_20090619`). ¿Qué hace el comando que se muestra en la línea 248?

Análisis de secuencias con EMBOSS - continuación

```
238 [user@server:~/dia1]$ fuzzpro TAIR9_pep_20090619 \
239 -pattern R-R-[ILV]-Y-D-[IAV]-[TVAL]-N-[VI]-[LF]  \
240 -rformat2 gff
241 Search for patterns in protein sequences
```

[22] `http://www.quantprime.de/`
[23] `http://www.expasy.ch/prosite/`
[24] `http://emboss.sourceforge.net/docs/themes/ReportFormats.html`
[25] 1000 pares de bases corriente arriba del sitio de inicio de la transcripción

```
Output report [at1g51370.fuzzpro]:
[user@server:~/dia1]$ fuzznuc -sequence TAIR9_upstream_1000_20090619 \
-pattern [AC]-[CT]-A-C-G-T-G-T-[AC] \
-rformat2 excel
Search for patterns in nucleotide sequences
Output report [at1g08520.fuzznuc]:
[user@server:~/dia1]$ grep -v "SeqName" at1g08520.fuzznuc | cut -f 1 | sort -u | wc -l
    1385
[user@server:~/dia1]$
```

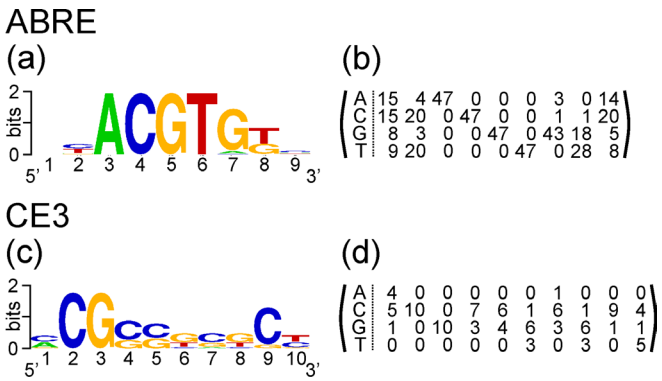


Fig. 2.7: Logos de secuencias de motivos ABRE y CE3 en promotores de genes vegetales.

Como ejercicio opcional puede crear una matriz en EMBOSS, y junto con el programa `profit`, buscar hits a la matriz en el mismo archivo de secuencias. De esta forma usted no pierde ninguna información sobre el perfil del sitio de regulación.

2.3.2. Alineamiento de secuencias

El alineamiento de secuencias consiste en encontrar las regiones o posiciones similares (evolutivamente relacionadas y por lo tanto homólogas) entre un par de, o mas, secuencias. La forma más sencilla e intuitiva de alinear secuencias es hacerlo gráficamente usando la técnica de "dot plot". El comando "`dottup`", crea un dot plot. Vamos a usar las secuencias del transcrito y el gen completo de AT4G01540.1; puede extraer las secuencias correspondientes de los archivos: TAIR9_cdna_20090619 y TAIR9_seq_20090619. Al ejecutar "dottup" (Línea251) obtenemos una representación gráfica de la estructura del gen (Figura 2.8). Como se dió cuenta, el dot plot es una representación gráfica de la similaridad entre pares de secuencias, cada una de las secuencias está representada por uno de los ejes en el dot plot, y un punto (dot) se dibuja cuando hay similaridad entre las dos secuencias. Regiones similares aparecen por lo tanto como líneas diagonales. Repeticiones aparecen como líneas diagonales en paralelo y deleciones o inserciones aparecen como secciones discontinuas entre diagonales.

El problema consiste en encontrar el alineamiento que maximize la similaridad entre las secuencias y en el caso de alineamientos pareados se puede resolver de forma sencilla y exacta dados ciertos parámetros, e.g., costo de gaps y matriz de similaridad usando el algoritmo de programación dinámica (Eddy, 2004a), que nos permite resolver el problema de alineamiento

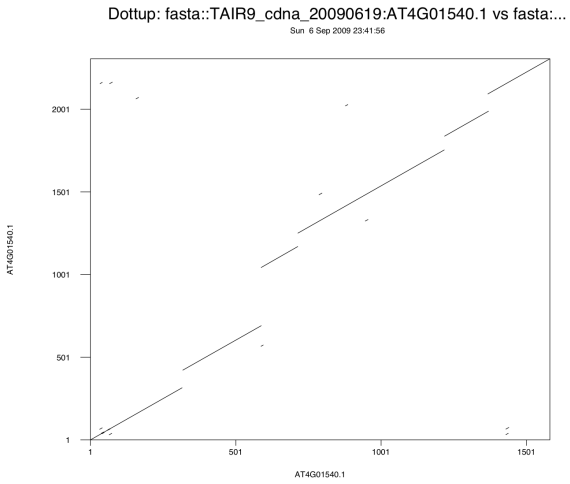


Fig. 2.8: Dot plot entre las secuencias del transcrito y el gen para AT4G01540.1

como una colección de sub-problemas, en donde cada uno de ellos se puede resolver independientemente de los demás.

En general un algoritmo de programación dinámica consiste de cuatro partes (Eddy, 2004a):

1. Una definicion recursiva del puntaje óptimo.
2. Una matriz de programación dinámica para recordar los puntajes óptimos de los sub-problemas, con el fin de resolver cada sub-problema una sola vez.
3. Un estrategia para llenar la matriz resolviendo los sub-problemas más pequeños primero.
4. Una forma de recorrer la matriz desde atrás hacia adelante recuperando la estructura de la solución óptima.

Y se lleva a cabo en 3 pasos:

1. Inicializar la matriz.
2. Llenar la matriz.
3. Rastreo de la matriz para encontrar la solución óptima.

Para cada una de las posiciones solo existen tres posibilidades, (i) las bases están alineadas, (ii) la base de la secuencias X está alineada a un gap, o (iii) la base de la secuencia Y está alineada a un gap. Cada una de esas posibilidades tiene un puntaje óptimo, que depende del costo de los gaps y de una matriz de similaridad entre bases.

A continuación vamos a desarrollar un ejemplo (tomado de Eddy, 2004a), alinearemos las secuencias `N: TGCTCGTA` y `M: TTCATA`. La matriz de similaridad (σ) que usamos es: +5 si las

bases son idénticas y -2 si las bases son diferentes. Y -6 puntos de penalización por cada gap (γ).

La definición recursiva de los puntajes óptimos la definimos como:

$$S_{(i,j)} = max \begin{cases} S_{(i-1,j-1)} + \sigma_{(x,y)} \\ S_{(i-1,j)} + \gamma \\ S_{(i,j-1)} + \gamma \end{cases} \tag{2.1}$$

El costo de alinear nada a nada $S_{(0,0)}$ es cero, que corresponde a la celda en la esquina superior izquierda de la matriz de programación dinámica, en rojo en el cuadro 2.1. Las fila $i = 0$ y la columna $j = 0$ corresponde a alinear cada secuencia a una cadena de gaps. La primera celda interesante es $j = 1; i = 1$ (en verde en el cuadro 2.1, el puntaje óptimo (aplicando la ecuación 2.1) para esta celda es:

$$S_{(1,1)} = max \begin{cases} 0 + 5 \\ -6 + (-6) \\ -6 + (-6) \end{cases} \tag{2.2}$$

Cuadro 2.1: Ejemplo de matriz de programación dinámica

		0	1	2	3	4	5	6	7	8
			T	G	C	T	C	G	T	A
0		**0**	**←-6**	**←-12**	**←-18**	**←-24**	**←-30**	**←-36**	**←-42**	**←-48**
1	T	**↑-6**	**↖ 5**							
2	T	**↑-12**								
3	C	**↑-18**								
4	A	**↑-24**								
5	T	**↑-30**								
6	A	**↑-36**								

Termine de llenar la matriz de programación dinámica. ¿Cuál es el alineamiento óptimo dada la matriz de similaridad y el costo de gaps?

La celda en el extremo inferior derecho nos da el puntaje del alineamiento óptimo. Confírmelo sumando los puntajes de cada una de las posiciones alineadas.

En el directorio `~/dia2/` hay programa de ejemplo (`global_eddy.c`) creado por Sean Eddy[26] que implementa un algoritmo de programación dinámica para el alineamiento global de secuencias. Ábralo con un editor de texto y modifique la matriz de similaridad y la penalización de gaps, compílelo de nuevo y vea como esto afecta el alineamiento final.

El algoritmo de programación dinámica descrito arriba encuentra el alineamiento global entre cualquier par de secuencias. Global en este contexto significa que el investigador supone que la similaridad entre las secuencias se extiende sobre toda su longitud. Si por el contrario el

[26] http://selab.janelia.org/

investigador supone que la región de similaridad entre las secuencias solo cobija una fracción de cada una de ellas, el alineamiento local es el apropiado. La principal diferencia con el algoritmo para alineamiento global, es que en este caso a todas las celdas con puntajes negativos se les asigna cero (Ecuación 2.3), y el rastreo de la matriz empieza con la celda de mayor puntaje hasta llegar a una celda con puntaje cero; de tal forma se puede encontrar mas de un alineamiento local entre un par de secuencias dado.

$$S_{(i,j)} = max \begin{cases} 0 \\ S_{(i-1,j-1)} + \sigma_{(x,y)} \\ S_{(i-1,j)} + \gamma \\ S_{(i,j-1)} + \gamma \end{cases} \tag{2.3}$$

En el caso de alineamientos de secuencias de proteínas los algoritmos son los mismos, solo cambia la matriz de "similaridad" entre residuos, y reciben el nombre de matrices de sustitución (Eddy, 2004b). Estas matrices de sustitución representan perfiles estadísticos de cambio de un amino ácido por otro. Use el comando que se muestra en la línea 255, para copiar la matriz de sustitución BLOSUM62 que viene con EMBOSS, la matriz de sustitución más popular. Los puntajes positivos representan sustituciones conservativas y los negativos sustituciones no conservativas. BLOSUM es una familia de matrices de sustitución, BLOSUM62 es la matriz que resultó de calcular las estadísticas de sustitución en alineamientos de referencia que tenían 62 % o menos de porcentaje de identidad. En forma similar BLOSUM80 y BLOSUM45 están calculadas usando alineamientos con menos de 80 % y 45 % de identidad, respectivamente (Henikoff and Henikoff, 1992).

En EMBOSS hay varios programas que permiten realizar alineamientos entre secuencias, use `wossname` para obtener una lista. Usaremos el programa "`stretcher`" para hacer un alineamiento global entre la secuencia del transcrito y el gen completo de AT4G01540.1, un factor de transcripción de la familia NAC en *A. thaliana*; puede extraer las secuencias correspondientes de los archivos: TAIR9_cdna_20090619 y TAIR9_seq_20090619, y luego ejecutar `stretcher`, o usar el comando directamente como se muestra en la línea 259; revise el archivo de salida usando `less`. ¿Puede identificar la estructura del gen? ¿Cuántos intrones tiene el gen?

Alineamiento de secuencias - continuación

```
251 [user@server:~/dia2]$ dottup TAIR9_cdna_20090619:AT4G01540.1 TAIR9_seq_20090619:AT4G01540.1
252 Displays a wordmatch dotplot of two sequences
253 Word size [10]:
254 Graph type [x11]:
255 [user@server:~/dia2]$ embossdata -fetch EBLOSUM62
256 Find and retrieve EMBOSS data files
257 File '/usr/local/share/EMBOSS/data/EBLOSUM62' has been copied successfully.
258 [user@server:~/dia2]$ less EBLOSUM62
259 [user@server:~/dia2]$ stretcher TAIR9_cdna_20090619:AT4G01540.1 TAIR9_seq_20090619:AT4G01540.1
260 Needleman-Wunsch rapid global alignment of two sequences
261 Output alignment [at4g01540.stretcher]:
262 [user@server:~/dia2]$ less at4g01540.stretcher
263 [user@server:~/dia2]$ est2genome TAIR9_cdna_20090619:AT1G01010.1 TAIR9_seq_20090619:AT1G01010.1
264 Align EST sequences to genomic DNA sequence
265 Output file [at1g01010.est2genome]:
```

```
266 [user@server:~/dia2]$ less  at1g01010.est2genome
267 [user@server:~/dia2]$ dottup TAIR9_pep_20090619:AT1G28470.1 TAIR9_pep_20090619:AT1G25580.1
268 Displays a wordmatch dotplot of two sequences
269 Word size [10]: 5
270 Graph type [x11]:
271 [user@server:~/dia2]$ water TAIR9_pep_20090619:AT1G28470.1 TAIR9_pep_20090619:AT1G25580.1
272 Smith-Waterman local alignment of sequences
273 Gap opening penalty [10.0]:
274 Gap extension penalty [0.5]:
275 Output alignment [at1g28470.water]:
276 [user@server:~/dia2]$ less at1g28470.water
277 [user@server:~/dia2]$
```

El programa "`est2genome`", le permite alinear secuencias de transcritos a secuencias genómicas. Alinee el transcrito de AT1G01010.1 contra el gen correspondiente (Línea 263)

Use el programa "`dottup`", variando el valor el tamaño de la ventana, para examinar la similaridad entre las proteínas NAC AT1G28470.1 y AT1G25580.1 (Línea 267). Como lo puede ver en su pantalla, las dos proteínas solo son similares en una región, i.e., localmente. En la línea 271 usamos el programa "`water`" para hacer un alineamiento local entre las mismas proteínas. Revise la salida del programa usando `less`.

2.3.3. BLAST: Basic Local Alignment Search Tool

Muchos de ustedes conocen la interfaz web de BLAST en el NCBI que se muestra en la Figura 2.10. En la primera parte de este tutorial vamos a hacer algunos ejercicios usando esta interfaz. En la segunda parte aprenderemos a usar BLAST en la línea de comandos con bases de datos locales.

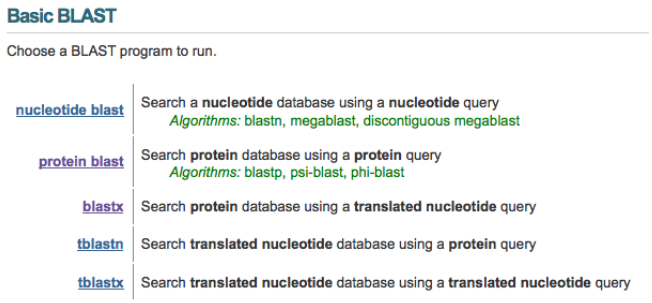


Fig. 2.9: Tipos de BLAST disponibles en el NCBI

En el directorio `~/dia2/` encuentra el archivo `desconocido.nuc.fa`, que contiene la secuencia de nucleótidos de un transcrito que usted descubrió al analizar la expresión diferencial de genes de *A. thaliana* en respuesta a luz ultravioleta (UV-A), tratamiento en el cual este transcrito era inducido. Copie la secuencia del transcrito y abra la página `http://blast.ncbi.nlm.nih.gov/` en el navegador Firefox. Vamos a realizar una búsqueda básica de BLAST, busque en la página una sección como la que aparece en la Figura 2.9 y seleccione la

opción `blastx`[27], ya que deseamos hacer una comparación de la secuencias de nuestro transcrito contra la base de datos de proteínas del NCBI.

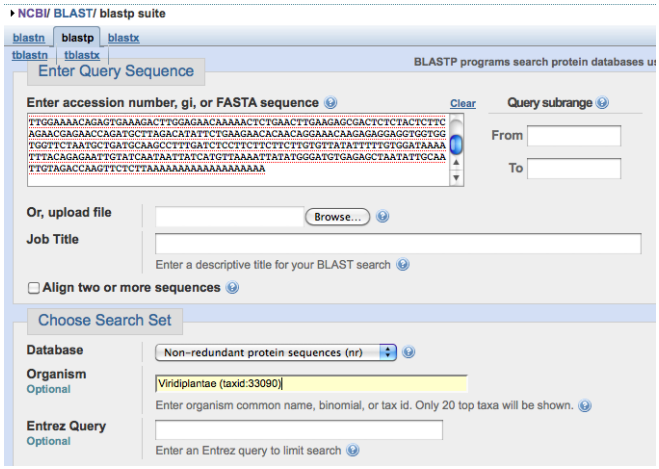


Fig. 2.10: Interfaz web de NCBI BLAST usando el programa blastx

En la página de `blastx` pegue su secuencia desconocida en el campo "**Enter query sequence**", escriba *Viridiplantae* en el campo "**Organism**", para restringir la búsqueda a las secuencias de plantas verdes (Figura 2.10). Asegúrese de que la base de datos seleccionada sea la base de datos no redundante de secuencias de proteínas.

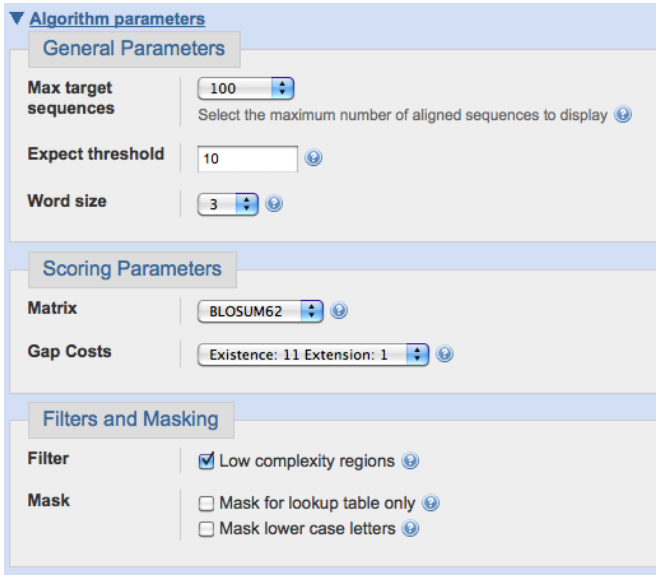


Fig. 2.11: Parámetros de búsqueda en BLAST

[27] ¿Por qué usar `blastx`?

Un poco más abajo, haga click en el vínculo "**Algorithm parameters**", lo que le mostrará la serie de opciones que se ven en la Figura 2.11. En la sección de "**General parameters**", encuentra el **Expected threshold** o **E value**. El E value es el número esperado de alineamientos con un puntaje igual o mayor al obtenido. En el momento de seleccionar los alineamientos importantes este es el parámetro más importante; como regla general alineamientos con E value menor que 1×10^{-5} representan secuencias homólogas. Sin embargo si está alineando secuencias muy cortas, 20 residuos, debe permitir alineamientos con E value muy alto, alrededor de 100. En la sección "**Scoring parameters**", puede seleccionar la matriz de sustitución (escoja BLOSUM80) y la penalización por introducir gaps en el alineamiento. Note que hay una diferencia entre el costo de introducir un gap y el de extenderlo ¿A qué se debe esa diferencia?

Asegúrese que la opción **Filter** en la sección **Filters and Masking** esté seleccionada, con el fin de reducir el número de alineamientos con secuencias no relacionadas evolutivamente.

Ahora presione el botón BLAST y espere por sus resultados.

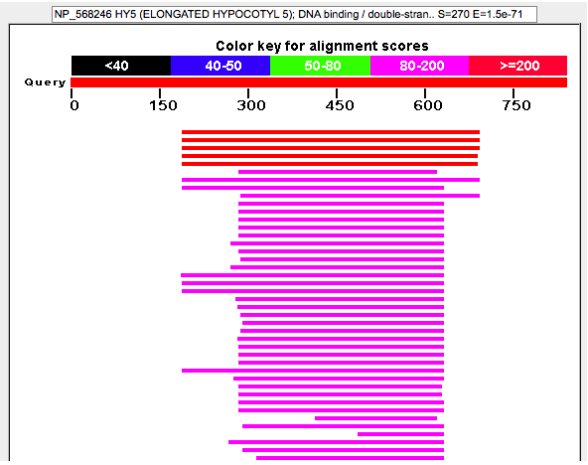


Fig. 2.12: Representación gráfica de los mejores alineamientos obtenidos en la búsqueda con `blastx`

En la parte superior de la página de resultado encuentra una gráfica como la que se ve en la Figura 2.12. Consiste en una representación de los mejores alineamientos con un código de colores que indica la longitud del alineamiento.

Un poco más abajo encuentra la lista de los mejores hits, donde se muestra el identificador de la secuencia hit, parte de su descripción, el puntaje del alineamiento entre su secuencia desconocida y la secuencia de la base de datos, y por último el E value que corresponde a ese hit.

La última parte de la sección de resultados está compuesta por los alineamientos propiamente dichos (Figura 2.14). Aquí va a encontrar nuevamente el puntaje y el E value del alineamiento. Adicionalmente, además del alineamiento, encuentra el número de posiciones en que las dos secuencias eran idénticas y similares (de acuerdo a la matriz de sustitución) y el número de gaps.

¿Puede decir algo sobre la función de su transcrito?

Sequences producing significant alignments:	Score (Bits)	E Value	
ref|NP_568246.1| HY5 (ELONGATED HYPOCOTYL 5); DNA binding / d...	270	1e-71	U G
gb|ABY83460.1| elongated hypocotyl 5 protein [Brassica rapa s...	220	2e-56	
ref|XP_002515537.1| transcription factor hy5, putative [Ricin...	201	9e-51	G
ref|XP_002324289.1| predicted protein [Populus trichocarpa] >...	201	9e-51	U G
ref|XP_002308656.1| predicted protein [Populus trichocarpa] >...	200	2e-50	U G
gb|AAO22523.1| HY5 [Brassica rapa subsp. pekinensis]	192	6e-48	
emb|CAN83322.1| hypothetical protein [Vitis vinifera]	190	3e-47	
sp|Q9SM50.1|HY5_SOLLC RecName: Full=Transcription factor HY5;...	179	4e-44	G
emb|CAO44204.1| unnamed protein product [Vitis vinifera]	173	3e-42	
gb|ACU17915.1| unknown [Glycine max]	170	2e-41	
gb|ACP28170.1| LONG1 [Pisum sativum] >gb|ACP28171.1| LONG1 [P...	170	2e-41	
gb|AAC05018.1| TGACG-motif-binding factor [Glycine max]	170	2e-41	G
gb|AAC05017.1| TGACG-motif binding factor [Glycine max] >gb|A...	170	2e-41	G
emb|CAA66478.1| transcription factor [Vicia faba var. minor]	166	5e-40	
ref|XP_002453510.1| hypothetical protein SORBIDRAFT_04g007060...	165	7e-40	U G
dbj|BAC20318.1| bZIP with a Ring-finger motif [Lotus japonicu...	163	3e-39	G
ref|XP_002437242.1| hypothetical protein SORBIDRAFT_10g023420...	159	5e-38	U G
ref|NP_001152483.1| transcription factor HY5 [Zea mays] >gb|A...	158	9e-38	U G
ref|NP_001046236.1| Os02g0203000 [Oryza sativa (japonica cult...	158	1e-37	U G
gb|EEC72704.1| hypothetical protein OsI_06291 [Oryza sativa I...	157	2e-37	
dbj|BAD15505.1| putative bZIP protein HY5 [Oryza sativa Japon...	157	2e-37	
gb|ABK23948.1| unknown [Picea sitchensis]	139	6e-32	
ref|NP_001058004.1| Os06g0601500 [Oryza sativa (japonica cult...	137	1e-31	U G
gb|EEC80926.1| hypothetical protein OsI_23604 [Oryza sativa I...	136	4e-31	
gb|ABK26016.1| unknown [Picea sitchensis]	128	1e-28	
ref|NP_001147637.1| transcription factor HY5 [Zea mays] >gb|A...	127	2e-28	U G
gb|EAY72732.1| hypothetical protein OsI_00597 [Oryza sativa I...	127	2e-28	

Fig. 2.13: Listado de "Hits"

>ref|NP_568246.1| U G HY5 (ELONGATED HYPOCOTYL 5); DNA binding / double-stranded DNA binding / transcription factor [Arabidopsis thaliana]

sp|O24646.1|HY5_ARATH G RecName: Full=Transcription factor HY5; AltName: Full=Protein LONG HYPOCOTYL 5; AltName: Full=bZIP transcription factor 56; Short=AtbZIP56

dbj|BAA21116.1| G HY5 [Arabidopsis thaliana]

dbj|BAA21327.1| G HY5 [Arabidopsis thaliana]

emb|CAB96661.1| G HY5 [Arabidopsis thaliana]

gb|ABF58937.1| G At5g11260 [Arabidopsis thaliana]

dbj|BAF01225.1| G bZip transcription factor HY5 / AtbZip56 [Arabidopsis thaliana]

Length=168

GENE ID: 830996 HY5 | HY5 (ELONGATED HYPOCOTYL 5); DNA binding / double-stranded DNA binding / transcription factor [Arabidopsis thaliana]

(Over 10 PubMed links)

Score = 216 bits (549), Expect = 3e-55
Identities = 168/168 (100%), Positives = 168/168 (100%), Gaps = 0/168 (0%)
Frame = +3

```
Query  192  MQEQATsslaasslpsssersssssapHLEIKEGIESDEEIRRVPEFGGEAVGKETSGRES  371
            MQEQATSSLAASSLPSSSERSSSSAPHLEIKEGIESDEEIRRVPEFGGEAVGKETSGRES
Sbjct  1    MQEQATSSLAASSLPSSSERSSSSAPHLEIKEGIESDEEIRRVPEFGGEAVGKETSGRES  60

Query  372  GSATGQERTQATVGESQRKRGRTPAekenkrlkrllrnrVSAQQARERKKAYLSELENRV  551
            GSATGQERTQATVGESQRKRGRTPAEKENKRLKRLLRNRVSAQQARERKKAYLSELENRV
Sbjct  61   GSATGQERTQATVGESQRKRGRTPAEKENKRLKRLLRNRVSAQQARERKKAYLSELENRV  120

Query  552  KDLENKNSELEERLSTLQNENQMLRHILknttgnkrgggggSNADASL  695
            KDLENKNSELEERLSTLQNENQMLRHILKNTTGNKRGGGGGSNADASL
Sbjct  121  KDLENKNSELEERLSTLQNENQMLRHILKNTTGNKRGGGGGSNADASL  168
```

>gb|ABY83460.1| elongated hypocotyl 5 protein [Brassica rapa subsp. rapa]

Length=167

Score = 173 bits (439), Expect = 2e-42

Fig. 2.14: Alineamientos resultantes de la búsqueda con `blastx`

La interfaz web de NCBI BLAST es muy amigable, pero tiene un par de problemas cuando trabajamos en genómica y proteómica, (i) no se pueden hace búsquedas contra bases de datos personalizadas o privadas y (ii) el número de secuencias que puede usar como **query** en cada búsqueda está restringido, normalmente no más de una secuencias por búsqueda. La alternativa

más poderosa para solucionar ambos problemas es instalar NCBI BLAST en un servidor local y configurar las bases de datos sobre las cuales se quiere realizar búsquedas. A continuación vamos a explotar una instalación local de NCBI BLAST, para (i) encontrar la región genómica en que está codificado un transcrito y (2) alinear ca. 10^5 secuencias cortas al genoma completo de *A. thaliana*.

Encontrando la región genómica de un transcrito.

Use la secuencia que se encuentra en el archivo `desconocido.nuc.fa` para hacer una búsqueda BLAST, usando `blastn` contra el genoma completo de *A. thaliana* como se muestra en la línea 278. Ya que BLAST realiza la búsqueda usando alineamientos locales, este resultado solo le dará una idea muy preliminar de la ubicación del transcrito en el genoma. Pero puede usar esta información para refinar la predicción del locus del transcrito usando `est2genome` de EMBOSS.

Usando NCBI BLAST en la línea de comandos

```
278 [user@server:~/dia2]$ blastall -p blastn -d TAIR9_chr_all.fa -i desconocido.nuc.fa -e 1e-5 \
279 -o desconocido.nuc.fa.blastout
280 [user@server:~/dia2]$ less desconocido.nuc.fa.blastout
281 [user@server:~/dia2]$ extractseq TAIR9_chr_all.fa:Chr5 -regions 'start-end'
282 Extract regions from a sequence
283 [user@server:~/dia2]$ est2genome desconocido.nuc.fa desconocido.nuc.fa.locuspre.fa -align
284 Align EST sequences to genomic DNA sequence
285 Output file [desconocido.est2genome]:
286 [user@server:~/dia2]$ less desconocido.est2genome
287 [user@server:~/dia2]$
```

Para saber que significa cada uno de los parámetros de `blastall` ejecute el programa sin ninguna opción.

Los resultados de esta búsqueda nos permiten concluir que el locus del transcrito está en el cromosoma número 5 de *A. thaliana*. ¿Cuáles son las coordendas en el cromosoma? Además el resultado de BLAST sugiere que el transcrito está conformado por lo menos por 4 exones. Vamos a usar este resultado como entrada para `est2genome`. Primero extraiga de la secuencia del cromosoma 5, la región detectada por BLAST adicionándole 5000 bp (pares de bases) corriente arriba y corriente abajo (Línea 281). Use `est2genome` para refinar la predicción del locus (Línea 283). ¿Qué ventajas ofrece usar `est2genome` comparado con un simple BLAST?

Alineando miles de secuencias en serie

Clark *et al.* (2007) han detectado miles de SNPs (Single Nucleotide Polymorphism) en el genoma de *A. thaliana* v7.0. Recientemente (Junio 2009) una nueva versión del genoma de este organismo fue liberada al público, en la cual algunos de los gaps todavía existentes en la secuencia de los cromosomas fueron resueltos, esto implica que es necesario mapear de nuevo los SNPs detectados por Clark *et al.* (2007) en la nueva versión del genoma. Afortunadamente los autores diseñaron un par de iniciadores de PCR para cada uno de los SNPs, cada cebador tiene 30 bp (pares de bases) y flanquea a la base polimórfica. Usted tiene que mapear una colección de 100000 SNPs en la nueva versión del genoma.

La suite de programas que viene con NCBI BLAST incluye `megablast`, que es una versión modificada de BLAST, optimizada para alinear secuencias que difieren en muy pocas posiciones. En este ejercicio, si usamos `blastn`, la búsqueda toma ca. 2h en una estación de trabajo con

procesador Intel Core 2 Duo de 2,13 GHz y 2GB en RAM, mientras que usando `megablast`, esta solo toma 26s ofreciendo los mismos resultados[28].

[28]Una opción mas sofisticada y eficiente consiste en emplear un arreglo de sufijos para representar la secuencia del genoma, usando Vmatch `http://www.vmatch.de/`

3

Análisis comparativo y evolución

Hace más de 35 años, el genético y biólogo evolutivo Theodosius Dobzhansky publicó uno de los artículos más referenciados en la historia de la biología (Dobzhansky, 1973). Quién no ha escuchado o leído la frase:

> Nothing in biology makes sense except in the light of evolution

Hoy en día, es más claro que nunca el poder del análisis comparado para la resolución de problemas biológicos. En este capítulo vamos a aprender algunas técnicas que nos permiten inferir la historia evolutiva de los genes.

3.1. Alineamiento múltiple

En teoría los algoritmos de programación dinámica que se describieron anteriormente para el caso de alineamientos pareados se pueden extender para el caso de un número arbitrario de secuencias. En la práctica, esto resulta muy costoso computacionalmente, por lo que se han desarrollado otros algoritmos que implementan atajos en la búsqueda de los alineamientos óptimos (heurísticas). El desarrollo de algoritmos para el alineamiento múltiple de secuencias es una de las áreas mas dinámicas de la bioinformática. Actualmente existen decenas de programas que implementan diferentes algoritmos (vea Notredame, 2007 y Lemey *et al.*, 2009 para una revisión reciente del tema).

En esta sección vamos a comparar los programas Muscle[1] y Mafft[2] que, de acuerdo a resultados de comparación con alineamientos de referencia, están entre los mejores y más rápidos.

Vamos a comparar las secuencias de proteínas de la familia de factores de transcripción E2F-DP[3] en los genomas de varias plantas (archivo `Plant_E2FDP.pep` en su directorio `~/dia4/`). Algunos de los programas que vamos a usar no aceptan identificadores de mas de 10 caracteres, por lo tanto vamos a renombrar nuestras secuencias usando el script `rename_seqs.pl`

[1] `http://www.drive5.com/muscle/`
[2] `http://align.bmr.kyushu-u.ac.jp/mafft/software/`
[3] `http://plntfdb.bio.uni-potsdam.de/v3.0/fam_mem.php?family_id=E2F-DP`

que se encuentra en su directorio (Línea 21). El archivo con la extensión `mapseq` almacena la equivalencia entre los identificadores originales y los nuevos, y el archivo con la extensión `modseq` contiene las secuencias con los nuevos identificadores. Ahora podemos realizar el primer alineamiento ejecutando el comando que se muestra en la línea 27. Con el fin de ahorrar un poco de tiempo el alineamiento con MUSCLE ha sido pre-calculado y lo encuentra en el archivo `Plant_E2FDP.pep.nr.modseq.muscle.aln.fa`. Ejecute el programa `JalView` y abra los dos archivo de alineamiento, dedique un tiempo a comparar los dos alineamientos (Figura 3.1).

Alineamiento múltiple de secuencias

```
1  [user@server:~/dia4]$ cd ~/dia4/
2  [user@server:~/dia4]$ ls -l Plant_E2FDP.pep
3  -rw-r--r-- 1 diriano gabi 29861 2009-09-14 18:32 Plant_E2FDP.pep
4  [user@server:~/dia4]$ ./nrdb.linux-x86 -d# Plant_E2FDP.pep > Plant_E2FDP.pep.nr
5
6
7  Progressive Statistics:
8
9             --------- Records ---------  -------------- Residues -----------
10 Database      Read  Duplicate  Written         Read   Duplicate     Written
11 Plant_E2FDP.pep      60        3         57       23,688        1042      22,646
12
13 Totals:         60        3         57       23,688        1042      22,646
14
15 No. of base word hits:  4 (4 total)
16 No. of 32-bit hash hits:  3
17 Total memory allocated:  0.250 MB
18 Longest comment line read:  145
19 Longest comment line written:  267
20 Longest sequence read:  875
21 [user@server:~/dia4]$ ./rename_seqs.pl Plant_E2FDP.pep.nr
22 [user@server:~/dia4]$ ls -l Plant_Sigma70-like.pep.fa*
23 -rw-r--r-- 1 diriano gabi  29861 2009-09-14 18:32 Plant_E2FDP.pep
24 -rw-r--r-- 1 diriano gabi  28832 2009-09-14 18:34 Plant_E2FDP.pep.nr
25 -rw-r--r-- 1 diriano gabi   6185 2009-09-14 20:36 Plant_E2FDP.pep.nr.mapseq
26 -rw-r--r-- 1 diriano gabi  23508 2009-09-14 20:36 Plant_E2FDP.pep.nr.modseq
27 [user@server:~/dia4]$ mafft --auto Plant_E2FDP.pep.nr.modseq > \
28 Plant_E2FDP.pep.nr.modseq.mafft.aln.fa
29 [user@server:~/dia4]$ JalView
```

Los dos alineamientos contienen regiones muy variables y en estas regiones los dos métodos generalmente no están de acuerdo. Asimismo, existen otras regiones bastante conservadas y allí los dos métodos ofrecen prácticamente los mismos resultados. En la mayoría de métodos de inferencia filogenética las columnas con gaps son ignoradas y es conveniente removerlas en el preprocesamiento de los datos. Usando JalView seleccione y remueva las columnas que contienen gaps en la mayoría de las secuencias. Asegúrese de mantener los bloques conservados en los que MUSCLE y MAFFT están de acuerdo. Salve el alineamiento modificado en formato fasta con el nombre `Plant_E2FDP.pep.nr.modseq.mafft.aln.fa.mod`.

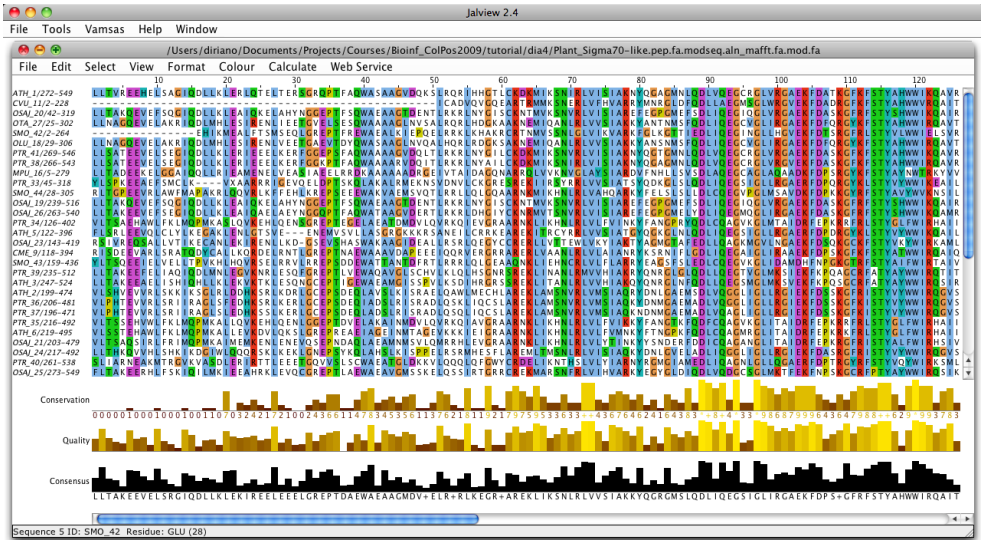


Fig. 3.1: Usando JalView para visualizar y editar alineamientos múltiples.

3.2. Inferencia de árboles filogenéticos

Estimando el modelo evolutivo

Los modelos evolutivos son un conjunto de suposiciones sobre el proceso de sustitución de nucleótidos o amino ácidos. El uso de un modelo u otro de evolución puede producir cambios en el análisis filogenético. Un buen modelo evolutivo debe ajustarse bien a los datos, en general los modelos más complejos se ajustan mejor a los datos que los modelos más simples, y esto debe tenerse en cuenta al seleccionar el modelo, i.e., el ajuste de los datos al modelo debe ser mucho mayor que el costo implícito de aumentar el número de parámetros del modelo. O como lo puso Abascal *et al.* (2005): la selección del modelo evolutivo se puede ver como una forma de identificar el modelo que, entre un conjunto de candidatos, es más cercano a la realidad, buscando un balance entre la precisión y la simplicidad.

Los dos programas más usados para la selección del modelo evolutivo son ModelTest para secuencias de DNA (Posada, 2006) y ProtTest para secuencias de amino ácidos (Abascal *et al.*, 2005). Como las nuestras son secuencias de proteínas usaremos ProtTest[4] para encontrar el modelo evolutivo al que mejor se ajustan nuestros datos. El archivo

```
Plant_E2FDP.pep.nr.modseq.mafft.aln.fa.mod.fa.prottest
```

tiene los resultados de este análisis, examínelo con el comando `less`

¿Cuántos modelos evolutivos fueron evaluados? ¿Que criterio se empleó para seleccionar el mejor modelo? ¿Cuál fue el modelo seleccionado?

[4] `http://darwin.uvigo.es/software/prottest.html`

Construyendo el árbol

PHYLIP es un paquete de programas para análisis filogenético. En este ejercicio usaremos los programas `seqboot`, `protdist`, `neighbor` y `consense` para estimar el árbol filogenético por el método de distancia[5].

Construyendo el árbol con PHYLIP

```
[user@server:~/dia4]$ seqboot
\ldots
[user@server:~/dia4]$ mv outfile Plant_E2FDP.pep.nr.modseq.mafft.aln.fa.mod.phy.seqboot
[user@server:~/dia4]$ protdist
\ldots
[user@server:~/dia4]$ mv outfile file.protdist
[user@server:~/dia4]$ neighbor
\ldots
[user@server:~/dia4]$ mv outfile file.neighbor.outfile
[user@server:~/dia4]$ mv outtre file.neigthbor.outtree
[user@server:~/dia4]$ consense
\ldots
[user@server:~/dia4]$ mv outfile file.consense.outfile
[user@server:~/dia4]$ mv outtree file.consense.outtree
[user@server:~/dia4]$ figtree textcolorbluetreefile
```

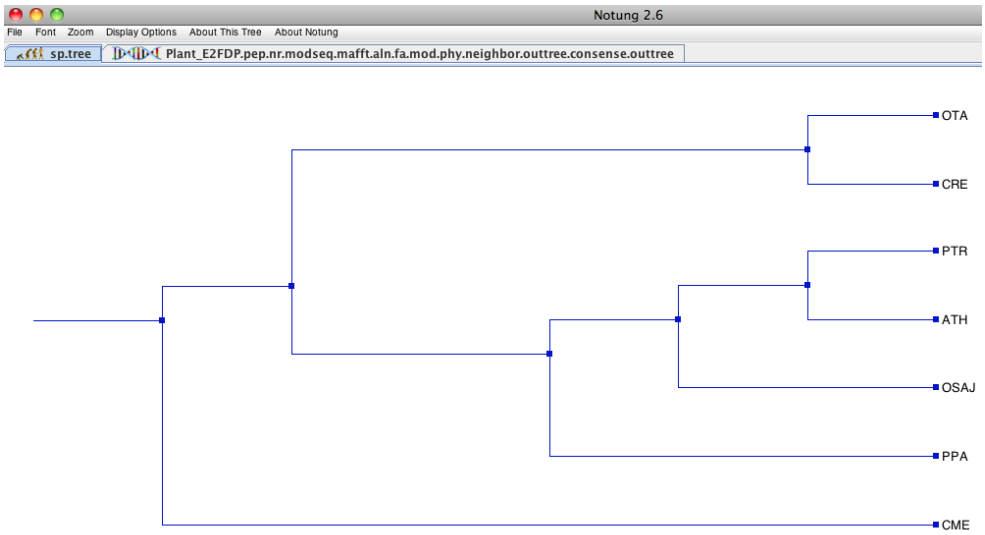


Fig. 3.2: Árbol de especies en Notung.

3.3. Reconciliar el árbol de genes con el árbol de especies, y definir de ortólogos/clanes

Las relaciones filogenéticas entre las especies que estamos estudiando son conocidas y se muestran en el archivo `sp.tree`, use `figtree` para visualizar ese árbol filogenético.

[5]Guía recomendada: `http://koti.mbnet.fi/tuimala/oppaat/phylip2.pdf`

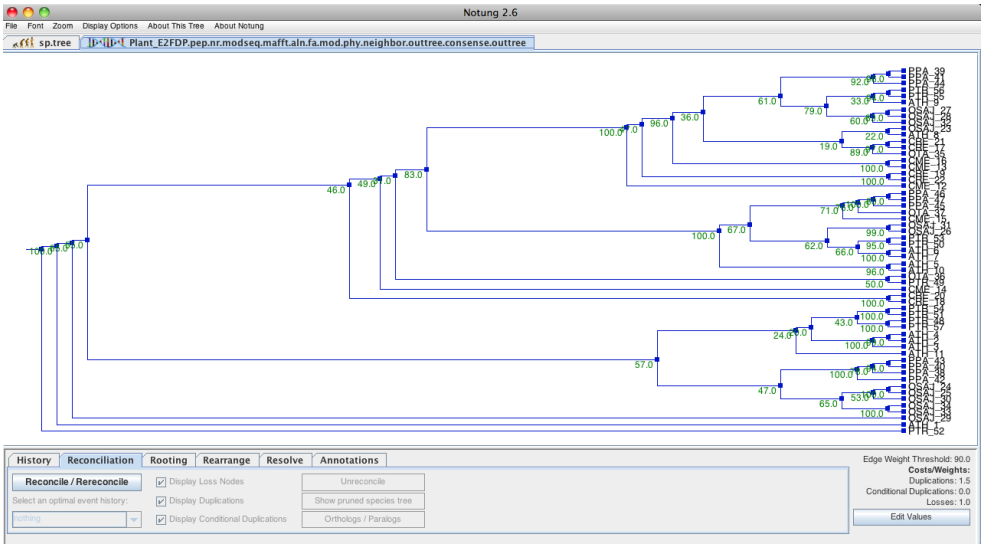


Fig. 3.3: Árbol de genes en Notung. En verde se muestran los valores de bootstrap.

Para poder establecer las relaciones entre genes ortólogos y parálogos (Fitch, 2000) es necesario establecer la raíz del árbol. Si esto no es posible, en lugar de establecer grupos de ortólogos se pueden identificar clanes (Wilkinson *et al.*, 2007).

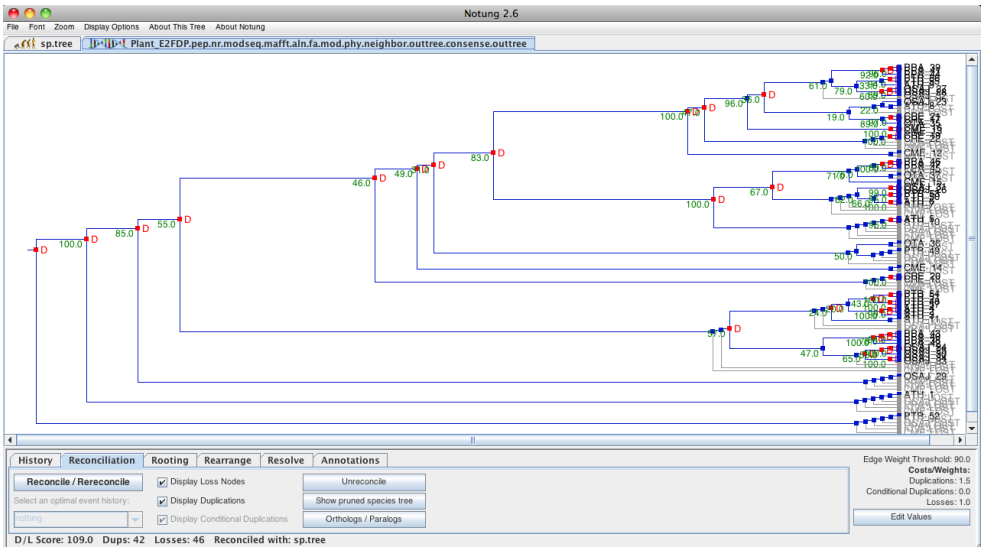


Fig. 3.4: Reconciliación en Notung.

Una estrategia para encontrar la raíz del árbol es reconciliar el árbol de genes con el árbol de las especies, si este último es conocido sin ambigüedad. Vamos a usar Notung[6] para reconciliar

[6] `http://www.cs.cmu.edu/~durand/Notung/`

los árboles e inferir las posibles raíces en el árbol de genes. en la Figura 3.2 se muestra el árbol de las especies. La Figura 3.3 muestra el árbol inferido en el paso anterior.

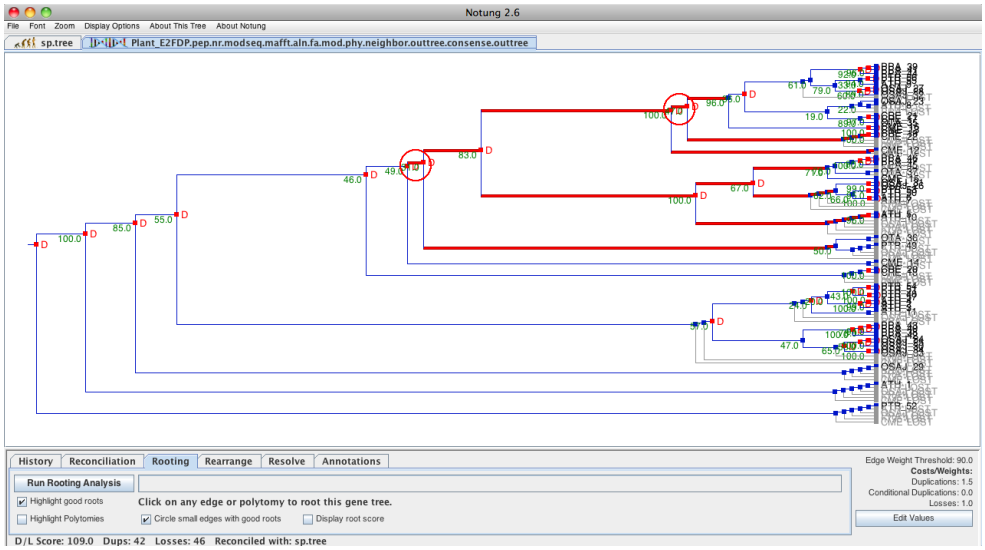


Fig. 3.5: Enraizado en Notung.

3.4. Identificación de motivos reguladores en regiones promotoras de genes ortólogos

Seleccione un par de ortólogos, o un par de genes dentro de un clan en el árbol creado en la sección anterior, uno de los genes tiene que ser de Arabidopsis y el otro de arroz (*Oryza sativa* ssp *japonica*. Use NCBI BLAST (web interface) para ubicar la región genómica de cada gen, y extraiga una región de ca. 1500 bp corriente arriba del sitio de inicio de la traducción, o del sitio de inicio de la transcripción si lo puede ubicar para cada gene (esto implicaría hacer la búsqueda BLAST con un flcDNA). Esta región de 1500 bp constituye nuestro promotor putativo y lo usaremos en dos tipos de análisis. Primero: Use MEME[7] para encontrar similaridades entre los promotores. Segundo: Use el servidor web FootPrinter 3.0[8] para comparar los promotores ortólogos. Compare los resultados de MEME y FootPrinter. Use TOMTOM[9] para comparar su motivo contra una base de datos de motivos conocidos.

[7] http://meme.sdsc.edu/meme4_1_1/cgi-bin/meme.cgi
[8] http://genome.cs.mcgill.ca/cgi-bin/FootPrinter3.0/FootPrinterInput2.pl
[9] http://meme.nbcr.net/meme4_1_1/cgi-bin/tomtom.cgi

4

Introducción a R - Parte I

Elizabeth González Estrada

4.1. Introducción

En este capítulo se presenta una breve introducción al uso del paquete R. Veremos los siguientes temas. Funciones básicas y paquetes, sistemas de comandos bajo Linux, importación y exportación de datos externos, y objetos R.

El paquete R es un ambiente para hacer procesamiento de datos, cálculos y gráficas. Algunas de las razones por las que muchos usuarios eligen R son las siguientes. Se distribuye sin costo[1]. Está disponible para los sistemas operativos más comunes (Windows, Unix y MacOS X). Permite manejar y almacenar grandes conjuntos de datos de manera eficiente. Tiene una gran flexibilidad para importar y exportar datos. Contiene un gran número de herramientas para hacer análisis estadístico de datos y gráficas de alta calidad, las cuales pueden ser exportadas a otras aplicaciones de manera sencilla. El usuario puede crear sus propias bibliotecas de funciones.

En la comunidad de bioinformática, R ha ganado popularidad gracias al proyecto Bioconductor[2] porque éste contiene muchas herramientas para hacer análisis de datos genómicos; por ejemplo, se pueden hacer análisis de microarreglos, de secuencias, etc.

Uso básico de R

Para iniciar una sesión de R en Linux escribimos R en la línea de comandos del programa Terminal. La pantalla de inicio de una sesión R es similar a la que se muestra en la Figura 4.1. En la última línea de esta figura aparece el símbolo del cursor precedido por el signo >. Esta línea es llamada la línea de comandos y en ella se escriben las instrucciones que queremos que R ejecute. Si una instrucción no cabe en una línea, el programa utilizará el signo + para indicar

[1] Se puede descargar desde la página del proyecto R `http://www.r-project.org`.
[2] `http://www.bioconductor.org/`

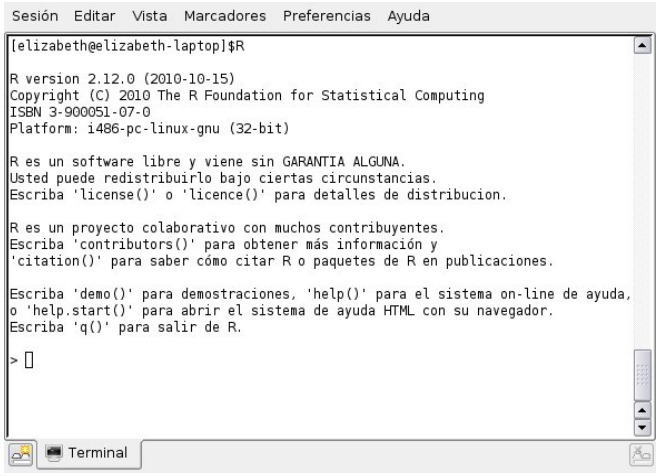


Fig. 4.1: Pantalla de inicio de R en Linux

que la instrucción continúa en la línea siguiente.

Para ejecutar un programa usamos el comando `source()`. Por ejemplo, para ejecutar el programa contenido en el archivo llamado prog1.txt que se encuentra en la carpeta `/home-/user/IntroR`, escribimos lo siguiente en la línea de comandos.

```
> source("prog1.txt")
```

Para salir de R, escribimos `q()`.

4.2. Funciones básicas y paquetes

Al instalar R se tiene acceso a muchas funciones para hacer cálculos básicos. Por ejemplo, se pueden calcular la media, mediana, suma, producto, raíz cuadrada, longitud, logaritmo, etc. de un conjunto de datos usando las funciones `mean, median, sum, prod, sqrt,length, log`, respectivamente. En el manual en línea de R[3] se puede encontrar una lista extensiva de las funciones disponibles.

Muchas funciones y conjuntos de datos se almacenan en paquetes, y para poder usarlas es necesario instalar el paquete que contiene las funciones que nos interesan y cargarlo en una sesión R. Enseguida se indica cómo instalar y cargar uno o más paquetes.

4.2.1. Instalación de paquetes

Hay varias formas de instalar un paquete en R. Aquí veremos cómo instalar un paquete desde la red exhaustiva de archivos R (CRAN) usando un ejemplo.

Para instalar el paquete llamado `multtest`, escribimos lo siguiente en la línea de comandos de R.

[3] `http://cran.at.r-project.org/doc/manuals/R-intro.html`

```
> install.packages("multtest")
```

Al ejecutar esta instrucción aparece un cuadro de diálogo en el que se nos pide que seleccionemos el lugar (CRAN mirror) desde el cual se hará la descarga del paquete. Es conveniente seleccionar el lugar más cercano. Note que el nombre del paquete se escribe entre comillas.

La sintaxis para instalar dos o más paquetes es la siguiente.

```
> install.packages(c("paquete1","paquete2",...))
```

Para usar un paquete es necesario cargarlo en la sesión usando la función `library()`. Para cargar el paquete `multtest`, escribimos:

```
> library(multtest)
```

Para ver una lista de los paquetes que están cargados en la sesión escribimos:

```
> search()
```

Al abrir R es conveniente establecer el directorio de trabajo. Para ver en qué directorio estamos trabajando escribimos:

```
> getwd()
```

El comando `dir()` muestra el contenido del directorio actual. Para definir a la carpeta `/home/user/IntroR` como nuestro directorio de trabajo, escribimos:

```
> setwd("/home/user/IntroR")
```

4.3. Sistemas de comandos bajo Linux

Los programas R se pueden ejecutar como archivos de procesamiento por lotes (BATCH) desde la línea de comandos del programa Terminal. Por ejemplo, para ejecutar el programa llamado prog1.txt, escribimos:

```
$ R CMD BATCH /home/user/IntroR/prog1.txt
```

Esta instrucción genera un archivo con extensión `.Rout` el cual lista los comandos que fueron ejecutados y sus salidas.

Para obtener más información sobre la ejecución de programas en modo BATCH, escribimos:

```
$ R CMD BATCH -help
```

4.4. Importación y exportación de datos

4.4.1. Importación de datos

Algunas funciones que permiten importar (leer) datos de otros programas a R son `read.delim` y `read.csv`.

La función `read.delim` permite pegar en R la información contenida en el portapapeles.

```
> read.delim("clipboard", header=T)
```

La opción `header=T` indica que las columnas tienen nombre.

Por medio de la función `read.csv` se puede leer el contenido de un archivo de datos separados por comas. Para leer los datos del archivo llamado `datos.csv`, usamos la instrucción:

```
> read.csv("datos.csv", header=T)
```

Cuando se desea importar una tabla que contiene campos vacíos y cadenas de caracteres largas, como es el caso de las tablas que contienen descripciones de genes, es conveniente usar la función `read.delim()`. Para leer los datos del archivo `datos.txt` hacemos lo siguiente:

```
> read.delim("datos.txt", na.strings= " ", fill=TRUE, header=T, sep="\t")
```

La opción `fill=TRUE` indica que se agreguen campos en blanco cuando las filas no tienen la misma longitud. Para obtener información sobre las opciones de la función `read.delim()` escribimos:

```
> ?read.delim
```


4.4.2. Exportación de datos

Algunas funciones que permiten exportar datos de R a otros programas son `write.table` y `write.csv`.

Para guardar en el portapapeles la información del conjunto de datos `iris`, el cual viene en la instalación básica de R, hacemos lo siguiente:

```
> write.table(iris, "clipboard", sep="\t", col.names=NA, quote=F)
```

Para guardar el conjunto de datos `iris` en un archivo de texto delimitado por tabulaciones, escribimos lo siguiente en la línea de comandos.

```
> write.table(iris, file="iris.txt", sep="\t", col.names = NA)
```

La opción `col.names = NA` asegura que los títulos estén alineados con las columnas cuando se exportan los nombres de filas e índices.

Para guardar el conjunto de datos `iris` en un archivo externo, escribimos:

```
> save(iris, file="iris.RData")
```

4.5. Objetos R

En R todo es considerado como un objeto que pertenece a una clase. En esta sección veremos los tipos de datos y objetos R más comunes (vectores, factores, listas, matrices y estructuras de datos). Revisaremos algunas funciones que permiten hacer operaciones básicas con objetos y veremos cómo extraer partes de objetos.

Existen diferentes tipos de datos en R:

- Datos numéricos: 1, 2, 3
- Caracteres: A, G, T, C
- Datos complejos: 1, b, 3
- Datos lógicos: TRUE, FALSE, TRUE

Para hacer comparaciones entre objetos se tienen los operadores: igual (==), desigual (! =), mayor/menor que (> <) y mayor/menor o igual que (>= <=). Para hacer operaciones lógicas se tienen los operadores: Y (&), O (|), No (!).

4.5.1. Vectores

Un vector es una colección ordenada de números, caracteres, datos complejos, valores lógicos.

Ejemplo: Suponga que los niveles de expresión genética de cierto gene de Juan, Pedro y Ana son 1, 1.5 y 1.25, respectivamente. Para capturar estos datos en R usamos la función `c()` (concatenar) de la siguiente forma.

```
> c(1,1.5,1.25)
```

La instrucción anterior genera un vector numérico. Para darle el nombre gene1 a este vector, escribimos:

```
> gene1 =  c(1,1.5,1.25)
```

Para ver el contenido de `gene1` escribimos:

```
> gene1
```

Para calcular la media, la varianza y la desviación estándar muestrales de `gene1` usamos las funciones `mean`, `var` y `sd` de la siguiente manera.

```
> mean(gene1)
> var(gene1)
> sd(gene1)
```

Alternativamente, podemos hacer lo siguiente.

```
> sum(gene1)/3
> v = sum((gene1-mean(gene1))**2)/2;v
> sqrt(v)
```

El cuadro 4.1 contiene ejemplos de diferentes tipos de vectores, de cómo crear un vector usando la función `sec()` y a partir de otros vectores.

Cuadro 4.1: Ejemplos de vectores

Ejemplo	Comentario
`> x = c(2,7,3,4)`	vector numérico
`> base = c('A","G","T",'C")`	vector de caracteres
`> y = c(1, "b", 3)`	vector lógico
`> sec1 = 1:10`	enteros del 1 al 10
`> sec2 = seq(10, 20, by=2)`	números $10, 12, ..., 20$
`> sec3 = c(sec1,sec2)`	vector formado por `sec1` y `sec2`
`> z = rep(base,3)`	repite 3 veces el vector `base`

Existen varias formas de seleccionar partes de un vector. En el cuadro 4.2 se dan algunos ejemplos.

Cuadro 4.2: Selección de partes de un vector

Ejemplo	Comentario
`> x[2:3]`	valores de las componentes 2 a 3 del vector x
`> x[-(2:3)]`	todo excepto las componentes 2 a 3 de x
`> x[x>3]`	valores de las componentes de x mayores que 3
`> x[1] = 9`	reemplaza el valor de la componente 1 por 9
`> month.name[1:4]`	meses de enero a abril
`> base == "G"`	vector con componentes FALSE y TRUE
`> which(z=="G")`	posición de G en el vector z
`> match(c("G","T"), z)`	posiciones de G y T en z

En el cuadro 4.3 se muestra el uso de algunas funciones útiles en el manejo de vectores.

Cuadro 4.3: Funciones útiles en el manejo de vectores

Función	Comentario
`> sort(x)`	ordena las componentes de x de menor a mayor
`> sort(base)`	ordena alfabéticamente las componentes de base
`> rev(sort(x))`	invierte el orden
`> order(x)`	posición de las componentes ordenadas en x
`> unique(z)`	componentes no duplicadas de z
`> sample(base, 5, replace=TRUE)`	selecciona una muestra de tamaño 5 con reemplazo del vector base

4.5.2. Factores

Un factor es un tipo especial de vector que contiene información de agrupación de sus componentes.

Ejemplo: Se tiene el siguiente vector.

```
> animal = c("perro", "gato", "mono", "perro", "perro", "gato")
```

Para crear un factor llamado `animalf` a partir del vector `animal` usamos la función `factor()`.

```
> animalf = factor(animal)
```

Para hacer una tabla de frecuencias para los niveles, escribimos:

```
> animalfr = table(animalf)
```

La función `tapply` se usa para aplicar una función (mean, median, sum, etc) a todos los niveles de un factor. Por ejemplo, suponga que se tienen los pesos correspondientes al vector `animal`.

```
> peso = c(102, 50, 150, 101, 103, 52)
```

Entonces se puede calcular la media de los valores de cada nivel de la siguiente forma.

```
> media = tapply(peso, animalf, mean)
```

4.5.3. Matrices y arreglos

Una matriz es una estructura bidimensional con datos del mismo tipo. Los arreglos pueden tener más de dos dimensiones.

Ejemplo: Suponga que se tienen los niveles de expresión de 3 genes más de Juan, Pedro y Ana.

```
> gene2 = c(1.35,1.55,1.00)
> gene3 = c(-1.10,-1.50,-1.25)
> gene4 = c(-1.20,-1.30,-1.00)
```

Se puede almacenar la información sobre los niveles de expresión de los 4 genes de Juan, Pedro y Ana en una matriz usando la función `rbind()` de la siguiente manera.

```
> rbind(gene1,gene2,gene3,gene4)
```

Alternativamente se puede usar la función `matrix()` como sigue.

```
> gendat = matrix(c(gene1,gene2,gene3,gene4), nrow=4, ncol=3, byrow=TRUE)
```

Al crear una matriz es conveniente asignarle nombres a las filas y a las columnas. Esto se hace como sigue.

```
> rownames(gendat) = c("gene1","gene2","gene3","gene4")
> colnames(gendat) = c("Juan","Pedro","Ana")
```

Para guardar la matriz `gendat` en un archivo de valores separados por comas con nombre gendat.csv, escribimos lo siguiente en la línea de comandos.

```
> write.csv(gendat,file="gendat.csv")
```

Para leer la matriz `gendat` usamos la función `read.csv`.

```
> read.csv("gendat.csv")
```

Alternativamente, se pueden usar las funciones `write.table` y `read.table`.

Con frecuencia interesa calcular medias y desviaciones estándares de filas o columnas de una matriz. La función `apply` aplica una función a cada columna (o fila) de una matriz.

Para calcular la media de los niveles de expresión de cada individuo escribimos

```
> apply(gendat,2,mean)
```

El número 2 indica que se va a calcular la media de cada columna de la matriz `gendat`. Para calcular la media por fila, es decir, la media de cada gene, escribimos lo siguiente.

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

Selección de partes de una matriz

Una matriz tiene dos índices [i,j], i se refiere a las filas y j a las columnas. Se pueden extraer partes de una matriz haciendo uso de los índices correspondientes a las partes de interés. En el cuadro 4.4 se dan algunos ejemplos.

Cuadro 4.4: Selección de elementos de una matriz

Ejemplo	Selección
`> gendat[1,2]`	segundo elemento de la primera fila
`> gendat[,1]`	primera columna (niveles de expresión de Juan)
`> gendat[1,]`	primera fila (gene1)
`> gendat[,c(2,3)]`	columnas 2 y 3
`> gendat[,c("Pedro",.^na")]`	columnas 2 y 3

Para seleccionar únicamente los genes de la matriz `gendat` que tienen un nivel de expresión promedio positivo hacemos lo siguiente. Primero calculamos la media de cada gene como sigue.

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

Entonces, los genes que tienen un nivel de expresión promedio positivo se obtienen así:

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

4.5.4. Listas

Una lista es más general que un vector, permite diferentes tipos de elementos.

```
> list("Human", 3000000000, 30000)
```

Se pueden dar nombres a los componentes de la lista de la siguiente forma.

```
> list(organism = "Human", genomeSizeBP= 3000000000, estGeneCount = 30000)
```

4.5.5. Estructuras de datos

Una estructura de datos es similar a una matriz, pero puede contener diferentes tipos de datos.

Ejemplo: Suponga que se tiene la siguiente información sobre los tamaños de los genomas de varios organismos.

```
> organismo  = c("Humano","Raton","MoscaF","GusanoR","Trigo")
> TamGenomaPB= c(3000000000,3000000000,135600000,97000000,12100000)
> estGeneCount = c(30000,30000,13061,19099,6034)
```

Una estructura de datos que contenga a estos vectores se crea como sigue.

```
> CompTamGenom = data.frame(organismo,TamGenomaPB,estGeneCount)
```

Los nombres de las columnas de esta estructura corresponden a los nombre de los vectores con que se formó. Para cambiar los nombres de `CompTamGenom` escribimos en la línea de comandos:

```
> names(CompTamGenom) = c("org", "tamano", "egc")
```

Extracción de partes de una estructura de datos

Para seleccionar la columna `org` de `CompTamGenom`, escribimos:

```
> CompTamGenom\$org
```

También podemos seleccionar la columna `org` así:

```
> CompTamGenom[,1]
```

En el primer caso, usamos el nombre de la columna para seleccionarla y en la segunda opción usamos su correspondiente índice.

La información del `gene1` se selecciona de la siguiente manera.

```
> CompTamGenom[1,]
```

Para seleccionar la información de los genes 1 y 3 de Pedro y Ana hacemos lo siguiente.

```
> CompTamGenom[c(1,3), c(2,3) ]
```

4.5.6. Información y manejo de objetos

Existen varias funciones que nos permiten manejar y obtener información de objetos.

Para listar los objetos creados durante la sesión podemos usar las funciones `ls()` u `objets()`. Para borrar un objeto, p. ej. para borrar el objeto `gene1`, escribimos:

```
> rm(gene1)
```

Si deseamos eliminar a `gene1` y `gene2`, escribimos:

```
> rm(gene1,gene2)
```

Para borrar todos los objetos creados durante la sesión sin que el paquete nos pregunte si queremos borrar todo, escribimos:

```
> rm(list = ls())
```

Para saber de qué tipo es un objeto usamos la función `class()`. Por ejemplo, para obtener el tipo de objeto de gendat escribimos la siguiente instrucción.

```
> class(gendat)
```

Si queremos obtener estadísticas descriptivas del objeto CompTamGenom, escribimos:

```
> summary(CompTamGenom)
```

5

Introduction to R - Part II

Adrian Alexa

5.1. The recycling rule

In a vectorized language like `R` a expression of the form `x + 1` is syntactically correct, and its a natural way to add the value `1` to each element of the vector `x`. However, in most of the cases there is a mismatch between the vector lengths. `x` has length n and `1` has length 1. In such a case, the *recycling rule* is applied: sorter vectors are recycled as often as needed such that the resulting vector will match the length of the longer one. In our case, vector `1` is repeated n times.

```
> x <- 1:10
> x + 1

 [1]  2  3  4  5  6  7  8  9 10 11
```

One can obtain the same effect as above in a more controlled way as follows:

```
> v1 <- rep(1, length(x))
> length(v1)

[1] 10

> x + v1

 [1]  2  3  4  5  6  7  8  9 10 11
```

The recycling rule applies to vectors of any lengths. In the following example

```
> v2 <- c(0, 1)
> x * v2 + 1
```

```
 [1]  1  3  1  5  1  7  1  9  1 11
```

the vector `v2`, which has length 2, its recycled five times, then its multiplied element by element with vector `x`. The result is added element wise to the vector obtained by recycling vector `1` ten times.

Fractional recycling its allowed in `R`, but a warning will be raised

```
> x + 1:3
```

```
 [1]  2  4  6  5  7  9  8 10 12 11
```

You should always pay attention to such warnings, since they indicate you are operating with different length vectors, and in most cases means you made an error!

5.2. Control structures

`R` offers typical control structures of the form `if-else`. Next we will make use of such structures to find out if there is at least one significant p-value in a vector containing, for example the p-values of 10 genes.

```
> set.seed(12)
> p.val <- runif(10)
> names(p.val) <- paste("gene", 1:length(p.val), sep = "")
```

Here we generated a named vector of 10 random numbers between 0 and 1. Please note the recycling rule is apply in the call of `paste` function.

```
> if (any(p.val <= 0.05)) {
+     print("There are significant p-values")
+ } else {
+     print("No significant p-values")
+ }
```

```
[1] "There are significant p-values"
```

In the first line the statement `p.val <= 0.05` is evaluated into a logical vector of the same length as vector `p.val`. Each element is either `TRUE` or `FALSE` if the respective element in `p.val` is smaller or equal than 0,05. Function `any()` is used to check if there is at least one element of the logical vector which is `TRUE`, in which case it returns value `TRUE`, otherwise it returns value `FALSE`. The `if-else` statement is checking the result of the function `any` and if it returns `TRUE`, it executes line 2 otherwise line 4.

We might want more than just the simple information on the existence of a significant p-value in our vector. We might be interested to label each gene with one of the two labels: "signif.ªnd "not-signif"depending on the respective p-value and the chosen cutoff. One could achieve this in various ways, but here we will make use of the `ifelse()` function.

```
> genes.sig <- ifelse(p.val <= 0.05, "signif", "not-signif")
> genes.sig

      gene1       gene2       gene3       gene4       gene5       gene6
"not-signif" "not-signif" "not-signif" "not-signif" "not-signif"    "signif"
      gene7       gene8       gene9      gene10
"not-signif" "not-signif"    "signif"    "signif"
```

The `ifelse()` function will create a new vector with the same length as its first argument, with the elements `"signif"` if the condition is true, otherwise `"not-signif"`. The `ifelse()` function is basically a vectorized version of `if-else` structure.

Another control statement is the `switch` function, which gives the option to make a decision based on multiple choices. A trivial example will be to compute, either the mean, median, or the number of significant p-values for our vector of p-values.

```
> stat <- "mean"
> switch(stat, mean = mean(p.val), median = median(p.val), numsig = sum(p.val <=
+     0.05))

[1] 0.3154036
```

Set `stat <- "numsig"` and run the `switch` statement again. How many significant genes are there?

5.3. Looping

Iteration in `R` is provided by the `for`, `while` and `repeat` statements. Additional to these statements the language semantics allows for iterator functions. We have seen an example of an iterator function in `ifelse`. Other important iterator functions are `apply`, `lapply` and `sapply`.

In this section we'll use a toy gene expression matrix (simulated example) to demonstrate to use of the control structures.

```
> set.seed(12)
> edat <- matrix(rnorm(20), nrow = 4, ncol = 5)
> dimnames(edat) <- list(paste("gene", 1:4, sep = ""), paste("sample",
+     1:5, sep = ""))
> edat

         sample1    sample2    sample3     sample4    sample5
gene1 -1.4805676 -1.9976421 -0.1064639 -0.77956651  1.1888792
gene2  1.5771695 -0.2722960  0.4280148  0.01195176  0.3405123
gene3 -0.9567445 -0.3153487 -0.7777196 -0.15241624  0.5069682
gene4 -0.9200052 -0.6282552 -1.2938823 -0.70346425 -0.2933051
```

Our gene expression matrix has four genes and five samples and is filled with normal-distributed random numbers. The first task is to compute the mean expression of each gene. We learned that we can access the i_{th} row of a matrix by using indexing.

```
> mean(edat[2, ])

[1] 0.4170705
```

Therefore one way to compute the mean of each gene would be to iterate over the rows of `edat` and use the function `mean` to compute the statistic. This can be achieved using the `for` statement

```
> result <- 0
> for (i in 1:nrow(edat)) result[i] <- mean(edat[i, ])
> names(result) <- rownames(edat)
> result

     gene1      gene2      gene3      gene4
-0.6350722  0.4170705 -0.3390522 -0.7677824
```

The `for` statement will repeat its body: `result[i] <- mean(edat[i, ])` for $i \in \{1, 2, 3, 4\}$. The result is stored in the `result` vector. After the loop ends we set the `names` attribute of the `result` vector to the row names of `edat`.

Even if the code above will correctly compute the gene-wise mean, there are two important issues which are usually ignored by the novice `R` user. The first and the really important problem is known as *pre-alocation of variables* or *growing objects*.

The problem stands in the first line where we set `result <- 0`. Here we tell `R` that `result` is a one dimensional vector and therefore `R` will allocate memory just for one element. However, as we iterate over the number of rows of `edat`, we change the size of the `result` vector; after the first iteration, the length of `result` will be 1, after the second iteration the length will be 2, and so on. Every time the size of the `result` vector is increased, `R` is allocating a new vector of right length and copy the content of the old vector into it. This is not a problem if we have few iterations or the size of `result` object is not to large. But once we get into hundred of iterations the for loop will be very slow resulting in inefficient code. The solution to this problem is to pre-allocate the vector (possible only when we know the length of the vector in advance).

```
> result <- numeric(nrow(edat))
> names(result) <- rownames(edat)
> i <- 1
> while (i <= nrow(edat)) {
+     result[i] <- mean(edat[i, ])
+     i <- i + 1
+ }
> result
```

```
     gene1      gene2      gene3      gene4
-0.6350722  0.4170705 -0.3390522 -0.7677824
```

In the above example we use the `while` statement to iterate over the matrix rows (try to use the `repeat` statement to achieve the same result). It should be mentioned that there is not performance difference between the three looping statements described above and the user should chose the one he feels more comfortable with.

The second issue with the first example, and to some degree with the second example, is that we need to "prepare"the vector which will store the results (pre-allocation, setting the names attributes, etc). In `R` a more elegant way to iterate over multidimensional vectors is to use a function from the `apply` family. These functions allow the user to apply a function/operator to each row or column of a matrix in a holistic manner.

```
> result.apply <- apply(edat, 1, mean)
> result.apply
```

```
     gene1      gene2      gene3      gene4
-0.6350722  0.4170705 -0.3390522 -0.7677824
```

The second argument of the function `apply` corresponds to the dimensions of the matrix: 1 means rows and 2 means columns. Also try `apply(edat, 2, mean)` to see the effect of the second argument. Please note that the names attributes are conserved in this case, which is a plus. One can be have a lot of flexibility with these family of functions as shown in the example bellow.

```
> apply(edat, 1, quantile, probs = 0.25)
```

If you want to compute the sum over each row or column you can do it even faster than `apply` using the following functions:

```
> rsum <- rowSums(edat)
> csum <- colSums(edat)
```

`R` also offers flow control structures. In the examples above we computed the row means, but now lets assume we don't allow a positive mean in the result. In the case we find a positive mean, we want to either stop the loop or to through an error message. These behaviours can be achieved using `break` and `next` statements.

```
> for (i in 1:nrow(edat)) {
+     r <- mean(edat[i, ])
+     if (r >= 0)
+         break
+     print(r)
+ }
```

```
[1] -0.6350722
```

We see that the loop is stopped as soon as a positive mean is found. Alternatively, if we just want to ignore a positive mean and we want to print only the negative means, then we need to use the `next` statement.

```
> for (i in 1:nrow(edat)) {
+     r <- mean(edat[i, ])
+     if (r >= 0)
+         next
+     print(r)
+ }
```

```
[1] -0.6350722
[1] -0.3390522
[1] -0.7677824
```

5.4. Functions

In the R language the users can create objects of mode function. These objects are true functions similar with the functions one would write in C, Java, Python, etc. The only difference here is that since the functions are objects, they can be passed as arguments to other functions, returned by functions or stored in complex object like lists for example.

Writing R functions provides the user with means of adding new functionality to the language and readability to his code. Here we define a function to compute a single sample t-test against the null hypothesis $\mu = 0$. The t-statistic for a vector $x = (x_1, ..., x_n)$ is given by

$$t = \frac{\bar{x}}{\sqrt{\frac{1}{n(n-1)} \sum_{j=1}^{n} (x_j - \bar{x})^2}},$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$. It is quite easy to implement this test statistic in R.

```
> tStat <- function(x) {
+     n <- length(x)
+     mx <- mean(x)
+     t <- mx/sqrt(sum((x - mx)^2)/(n * (n - 1)))
+     return(t)
+ }
```

As we can see, one defines the body of the function and assigns it to a variable, in this case `tStat`. Our function has one argument denoted by `x`. Every function must return an object, and any type of objects can be returned by a function. In our case the function will return a scalar (vector of dim 1), namely the t-statistic.

Once the function is defined the user can use it like any other predefined R function.

```
> tStat(c(0.4, -0.23, 1, 3, 4, 2))

[1] 2.577394
```

Since functions are objects, they can passed as arguments to other functions. We have seen such an example with the `apply` family of functions. In a similar fashion we can use the `tStat` function as a parameter for the `apply` iterator. Here we would like to compute the gene-wise t-statistic for the toy gene expression matrix `edat`.

```
> gene.tstat <- apply(edat, 1, tStat)
> str(gene.tstat)

 Named num [1:4] -1.14 1.32 -1.32 -4.64
 - attr(*, "names")= chr [1:4] "gene1" "gene2" "gene3" "gene4"
```

5.5. Graphics

R has extensive graphical abilities. In this section we will briefly go through the most basic of them.

The main R graphics function is `plot`. In addition to `plot` there are functions for adding points and lines to existing graphs, for placing text at specified positions, for specifying tick marks and tick labels, for labelling axes, and so on.

Lets first plot the square root of the first 50 integers.

```
> x <- 1:50
> plot(x, sqrt(x))
```

The `plot` function is generating a scatterplot. To this plot we could add further points using the `points` function. Lets say we want to mark red the point $(x, y) = (10, sqrt(10))$ and also add a vertical line at $x = 25$

```
> points(10, sqrt(10), col = 10, pch = 19)
> abline(v = 25, col = 5, lwd = 3)
```

We would might also wish to draw the $\log(x)$ curve on the current plot. This can be achieved using the `lines` function.

```
> lines(x, log(x), col = "blue")
```

Any call of the `plot` function will erase the current graphical device. For example, we can plot the $\log(x)$ curve by just calling `plot`

```
> plot(x, log(x), type = "l", xlab = "x", ylab = "log(x)")
```

```
> hist(c(rnorm(1000, mean = 3, sd = 2), rnorm(1000, mean = -1)), 50, xlab = expression(x ==
+     delta * N(3, 2) + (1 - delta) * N(-1, 1)), main = "Histogram of a Gaussian mixture")
```

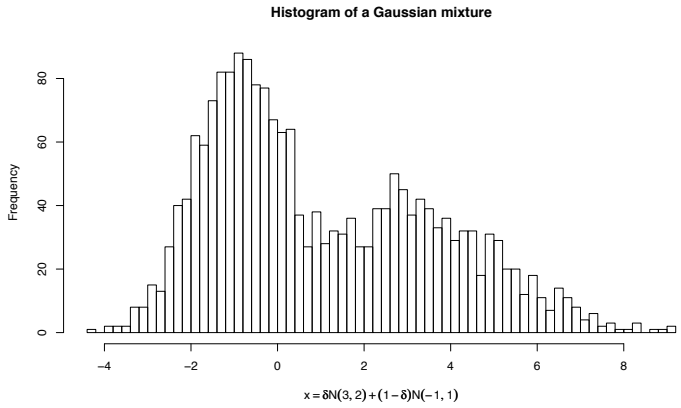


Fig. 5.1: The histogram for a mixture of two Gaussians.

We see that the range of the y axis changed, more exactly it is automatically set to fit the range of the y values, in our case log(1) and log(50) respectively. Also notice that using the `xlab` and `ylab` parameters we set the labels of the corresponding axes.

A very useful plotting function is `hist` which will plot the sample distribution or histogram.

In Figure 5.1 we plotted the histogram of a mixture model. The argument `main` can be used in most of the plotting functions and its used to set the plot title. We also made use of the `expression` function to use a formula as the x axis label.

Another useful plotting function is `matplot`. The function can be used to plot each column of a matrix. This can be very useful when we deal with gene expression data. If we transpose the matrix then we can plot each gene expression profile. For our toy matrix we have:

In Figure 5.2 we changed the labels of the x axis to match the column names of out matrix. In order to achieve this we first need to tell the `plot` function not to draw any axis, see `axes = FALSE` argument in the call of `matplot` function. Then we plot the default y axis and we set the box around the plot. The user can control how each axis should look like using the `axis` function. The first parameter determines which axis we are modifying.

As in the case of `plot` function one can use the `matpoints` function to add additional columns to the existing device.

```
> matpoints(rep(0, ncol(edat)), type = "b", lty = 1, lwd = 3, pch = 19)
```

Multiple plots in one figure Graphical parameters can be set using the `par` function. One can see all the current values of the parameters by calling this function without an argument (the function returns a list).

```
> par()
```

```
> matplot(t(edat), type = "l", lty = 1, ylab = "Expression value", axes = FALSE)
> axis(2)
> box()
> axis(1, at = 1:5, label = colnames(edat))
```

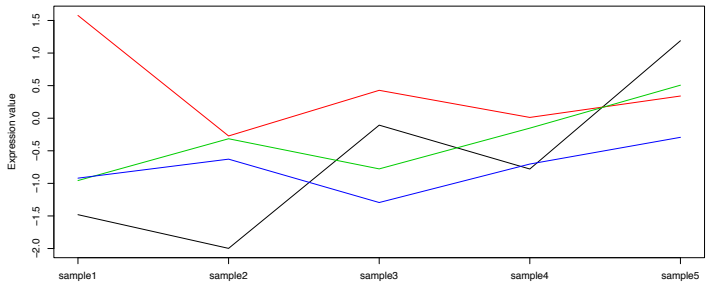


Fig. 5.2: Genes profiles for our toy example.

The `mfrow` and `mfcol` are some of the most useful parameters within `par`. Next we will plot the expression profile of each gene in a separate plotting area, but we want to have all four plots in the same figure.

Try to replace `mfcol` with `mfrow` and see what happens.

Saving plots `R` allows the user to save the plots either in a bitmap format (bmp, jpeg, png) or in a vectorial format (pdf, ps). The user needs to tell `R` in which format he wants to save his plot. Lets assume we want to save the plot generated using `matplot` into a PDF file.

```
> pdf(file = "Expression_Profiles.pdf")
> matplot(t(edat), type = "l", lty = 1, ylab = "Expression value", axes = FALSE)
> axis(2)
> box()
> axis(1, at = 1:5, label = colnames(edat))
> matpoints(rep(0, ncol(edat)), type = "b", lty = 1, lwd = 3, pch = 19)
> dev.off()
```

You will find the saved file in the current working directory (use `getwd()`). Please note that the last line calls the function `dev.off`. It is very important not to omit this line once you finished the code chunk that is producing the plot. If omitted, the file we want to save our plot is not closed and the plot is not completely saved.

Its always better to save your plots into a vectorial format, since this format allows scaling without quality loss. For the complete list of available graphical devices use `help("Devices")`.

```
> r <- range(edat)
> par(mfcol = c(2, 2))
> for (i in 1:nrow(edat)) plot(1:ncol(edat), edat[i, ], type = "l", ylim = r,
+     xlab = rownames(edat)[i], ylab = "expression")
```

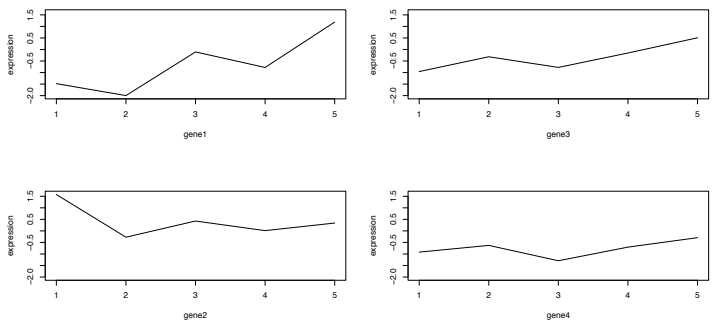


Fig. 5.3: Genes profiles for our toy example in separate plots

5.6. Performance issues

Checking the time your code takes to execute it is a very good way to improve your R skills and also to look for alternative ways of carrying out computations. The easiest way to achieve this is via `system.time` function.

```
> s <- 0
> x <- runif(1e+05)
> system.time({
+     for (i in 1:length(x)) s <- s + x[i]
+ })
```

```
   user  system elapsed
  0.196   0.000   0.225
```

Here we use a `for` loop to compute the sum of the elements of vector `x` (not recommended!). Compare this running time with the time the `sum` function takes:

```
> system.time({
+     s2 <- sum(x)
+ })
```

```
   user  system elapsed
  0.004   0.000   0.009
```

Once you know that a chunk of code takes a long time, you might want to know which part of your function is consuming most of the time. In R this can be achieved using the `Rprof` function.

```
> nr <- 50000
> nc <- 1000
> Rprof(interval = 0.01, memory.profiling = TRUE)
> mat <- matrix(integer(1), nrow = nr, ncol = nc)
> for (i in 1:nr) mat[i, ] <- sample(nc)
> Rprof(NULL)
> summaryRprof(memory = "both")
```

5.7. Summary

Some points that are worth remembering about R:

- Speedup is possible in R using a few tricks. Sometimes are easy to implement, sometimes they are not.
- Start with simple approaches with witch you feel comfortable. Once the code is functional you can start the optimization.
- The choice of data structure is critical to the performance. Use vectorization whenever its possible.
- Do not grow objects and avoid recopying large objects.
- `for` loops are not slow, but the operations performed inside are. Have a compromise between `for` loops and `apply`. Do not over do it by just using `apply`!

If you have any comments, criticisms or suggestions on this tutorial, please contact me: alexa@mpi-inf.mpg.de

6

Exploratory Data Analysis — Clustering gene expression data

Adrian Alexa

We consider expression data from patients with acute lymphoblastic leukemia (ALL) that were investigated using HGU95AV2 Affymetrix GeneChip arrays [Chiaretti et al., 2004]. The data were normalized using quantile normalization and expression estimates were computed using RMA. The preprocessed version of the data is available in the Bioconductor data package `ALL` from `http://www.bioconductor.org`. Type in the following code chunks to reproduce our analysis. At each ">" a new command starts, the "+" indicates that one command spans over more than one line.

```
> library(ALL)
> data(ALL)
> help(ALL)
> cl <- substr(as.character(ALL$BT), 1, 1)
> dat <- exprs(ALL)
```

The data consist of 128 samples with 12625 genes. The most pronounced distinction in the data is between B- and T-cells. You find these true labels in the vector `cl`. In this session we try to rediscover the labels in an unsupervised fashion.

6.1. Hierarchical clustering of samples

The first step in hierarchical clustering is to compute a matrix containing all distances between the objects that are to be clustered. For the ALL data set, the distance matrix is of size 128×128. From this we compute a dendrogram using complete linkage hierarchical clustering:

```
> d <- dist(t(dat))
> image(as.matrix(d))
```

```
> hc <- hclust(d, method = "complete")
> plot(hc, labels = cl)
```

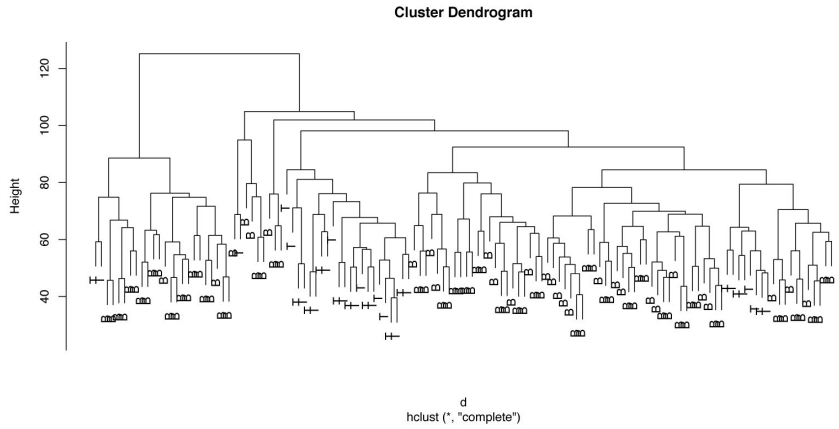


Fig. 6.1: Dendrogram of complete linkage hierarchical clustering. The two cell types are not well separated.

We now split the dendrogram into two clusters (using the function `cutree`) and compare the resulting clusters with the true classes. We first compute a contingency table and then apply an independence test.

```
> groups <- cutree(hc, k = 2)
> table(groups, cl)
```

```
      cl
groups  B  T
     1 73 31
     2 22  2
```

```
> fisher.test(groups, cl)$p.value
```

```
[1] 0.03718791
```

The null-hypothesis of the Fisher test is "the true labels and the cluster results are independent". The p-value shows that we can reject this hypothesis at a significance level of 5 %. But this is just statistics: What is *your* impression of the quality of this clustering? Did we already succeed in reconstructing the true classes?

Exercise: In the lecture you learned that hierarchical clustering is an agglomerative bottom-up process of iteratively joining single samples and clusters. Plot `hc` again, using as parameter `labels=1:128`. Now, read the help text for `hclust`: `hc$merge` describes the merging of clusters. Read the details carefully. Use the information in `hc$merge` and the cluster plot to explain the clustering process and to retrace the algorithm.

Exercise: Cluster the data again using `single` linkage and `average` linkage. Do the dendrogram plots change?

6.2. Gene selection before clustering samples

The dendrogram plot suggests that the clustering can still be improved. And the p-value is not *that* low. To improve our results, we should try to avoid using genes that contribute just noise and no information. A simple approach is to exclude all genes that show no variance across all samples. Then we repeat the analysis from the last section.

```
> genes.var <- apply(dat, 1, var)
> genes.var.select <- order(genes.var, decreasing = T)[1:100]
> dat.s <- dat[genes.var.select, ]
> d.s <- dist(t(dat.s))
> hc.s <- hclust(d.s, method = "complete")
> plot(hc.s, labels = cl)
```

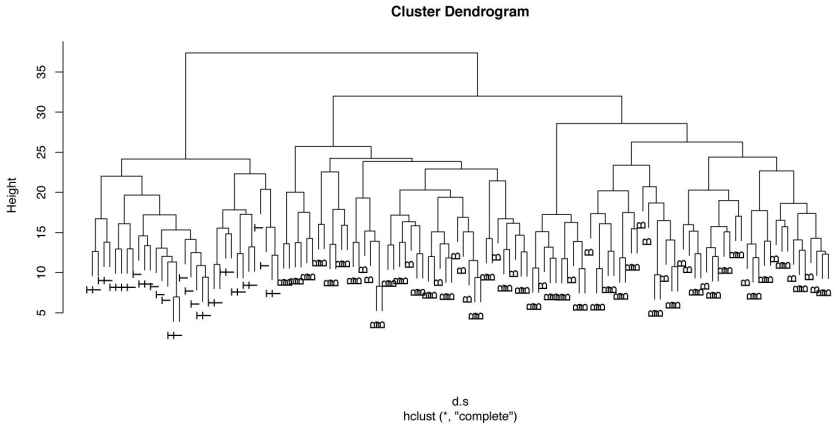


Fig. 6.2: Dendrogram of hierarchical clustering on 100 genes with highest variance across samples. The two classes are clearly separated.

Plot the distance matrix `d.s`. Compare it to `d`. Then split the tree again and analyze the result by a contingency table and an independence test:

```
> groups.s <- cutree(hc.s, k = 2)
> table(groups.s, cl)
```

```
        cl
groups.s  B  T
       1 95  0
       2  0 33
```

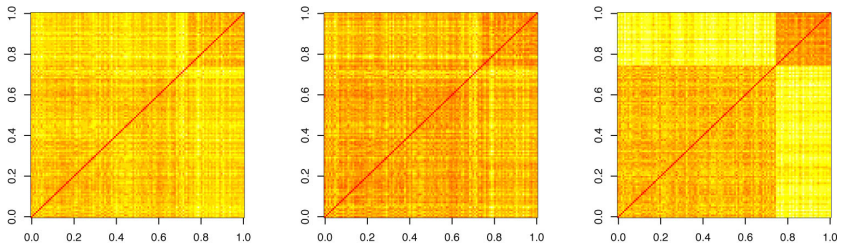


Fig. 6.3: Distance matrices for 128 samples, using all genes (left), 100 random genes (middle), and 100 high-variance genes (right).

```
> fisher.test(groups.s, cl)$p.value
```

```
[1] 2.326082e-31
```

It seems that reducing the number of genes was a good idea. But where does the effect come from: Is it just dimension reduction (from 12625 to 100) or do high-variance genes carry more information than other genes? We investigate by comparing our results to a clustering on 100 *randomly* selected genes:

```
> genes.random.select <- sample(nrow(dat), 100)
> dat.r <- dat[genes.random.select, ]
> d.r <- dist(t(dat.r))
> image(as.matrix(d.r))
> hc.r <- hclust(d.r, method = "complete")
> plot(hc.r, labels = cl)
> groups.r <- cutree(hc.r, k = 2)
> table(groups.r, cl)
> fisher.test(groups.r, cl)$p.value
```

Plot the distance matrix `d.r` and compare it against the other two distance matrices `d` and `d.s`. Repeat the random selection of genes a few times. How do the dendrogram, the distance matrix, and the p-value change? How do you interpret the result?

Finally, we compare the results on high-variance and on random genes in a cluster plot in two dimensions:

```
> library(cluster)
> par(mfrow = c(2, 1))
> clusplot(t(dat.r), groups.r, main = "100 random genes", color = T)
> clusplot(t(dat.s), groups.s, main = "100 high-variance genes", color = T)
```

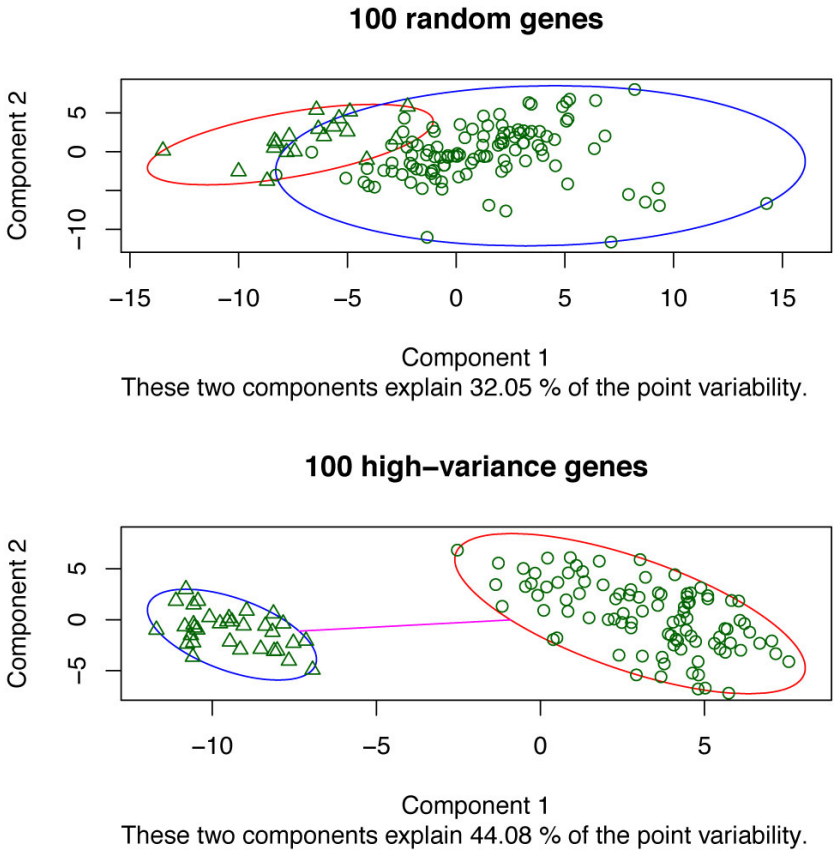


Fig. 6.4: You see the projection of the data onto the plane spanned by the first two principal components (= the directions in the data which explain the most variance). The group labels result from hierarchical clustering on 100 genes. In the upper plot the genes are selected randomly, in the lower plot according to highest variance across samples. The red and blue ellipses indicate class boundaries. Do we need both principal components to separate the data using high-variance genes?

6.3. Partitioning methods

In hierarchical methods, the samples are arranged in a hierarchy according to a dissimilarity measure. Extracting clusters means cutting the tree-like hierarchy at a certain level. Increasing the number of clusters is just subdividing existing clusters – clusters are nested.

In partitioning methods this is different. Before the clustering process you decide on a fixed number `k` of clusters, samples are then assigned to the cluster where they best fit in. This is more flexible than hierarchical clustering, but leaves you with the problem of selecting `k`. Increasing the number of clusters is not subdivision of existing clusters but can result in a totally new allocation of samples. We discuss two examples of partitioning methods: *K-means* clustering and *PAM* (Partitioning Around Medoids).

6.3.1. k-means

First look at `help(kmeans)` to become familiar with the function. Set the number of clusters to `k=2`. Then perform k-means clustering for all samples and all genes. k-means uses a random starting solution and thus different runs can lead to different results. Run k-means 10 times and use the result with smallest within-cluster variance.

```
> k <- 2
> withinss <- Inf
> for (i in 1:10) {
+     kmeans.run <- kmeans(t(dat), k)
+     print(sum(kmeans.run$withinss))
+     print(table(kmeans.run$cluster, cl))
+     cat("----\n")
+     if (sum(kmeans.run$withinss) < withinss) {
+         result <- kmeans.run
+         withinss <- sum(result$withinss)
+     }
+ }
> table(result$cluster, cl)
```

The last result is the statistically best out of 10 tries – but does it reflect biology? Now do k-means again using the 100 top-variance genes. Compare the results.

```
> kmeans.s <- kmeans(t(dat.s), k)
> table(kmeans.s$cluster, cl)
```

6.3.2. PAM: Partitioning Around Medoids

As explained in today's lecture, PAM is a generalization of k-means. Look at the help page of the R-function. Then apply `pam` for clustering the samples using all genes:

```
> result <- pam(t(dat), k)
> table(result$clustering, cl)
```

Now use `k=2:50` top variance genes for clustering and calculate the number of misclassifications in each step. Plot the number of genes versus the number of misclassifications. What is the minimal number of genes needed for obtaining 0 misclassifications?

```
> ngenes <- 2:50
> o <- order(genes.var, decreasing = T)
> miscl <- NULL
> for (k in ngenes) {
+     dat.s2 <- dat[o[1:k], ]
```

```
+     pam.result <- pam(t(dat.s2), k = 2)
+     ct <- table(pam.result$clustering, cl)
+     miscl[k] <- min(ct[1, 2] + ct[2, 1], ct[1, 1] + ct[2, 2])
+ }
> xl = "# genes"
> yl = "# misclassification"
> plot(ngenes, miscl[ngenes], type = "l", xlab = xl, ylab = yl)
```

6.4. How many clusters are in the data?

Both `kmeans` and `pam` assume that the number of clusters is known. The methods will produce a result no matter what value of `k` you choose. But how can we decide, which choice is a good one?

6.4.1. The objective function

Partitioning methods try to optimize an objective function. The objective function of k-means is the sum of all within-cluster sum of squares. Run k-means with `k=2:20` clusters and 100 top variance genes. For `k=1` calculate the total sum of squares. Plot the obtained values of the objective function.

```
> totalsum <- sum(diag((ncol(dat.s) - 1) * cov(t(dat.s))))
> withinss <- numeric()
> withinss[1] <- totalsum
> for (k in 2:20) {
+     withinss[k] <- sum(kmeans(t(dat.s), k)$withinss)
+ }
> plot(1:20, withinss, xlab = "# clusters", ylab = "objective function", type = "b")
```

Why is the objective function not necessarily decreasing? Why is `k=2` a good choice?

6.4.2. The silhouette score

First read the *Details* section of `help(silhouette)` for a definition of the silhouette score. Then compute silhouette values for PAM clustering results with `k=2:20` clusters. Plot the silhouette widths. Choose an optimal `k` according to the maximal average silhouette width. Compare silhouette plots for `k=2` and `k=3`. Why is `k=2` optimal? Which observations are misclassified? Which cluster is more compact?

```
> asw <- numeric()
> for (k in 2:20) {
+     asw[k] <- pam(t(dat.s), k)$silinfo$avg.width
+ }
```

```
> plot(1:20, asw, xlab = "# clusters", ylab = "average silhouette width", type = "b")
> plot(silhouette(pam(t(dat.s), 2)))
```

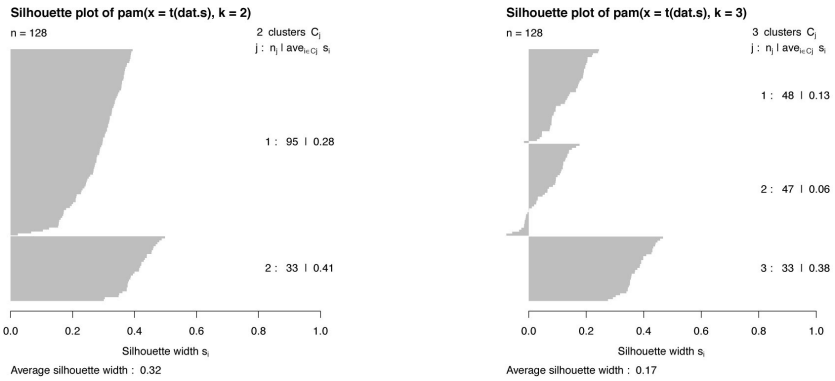


Fig. 6.5: Silhouette plot for `k=2` and `k=3`

6.5. How to check significance of clustering results

Randomly permute class labels 20 times. For `k=2`, calculate the average silhouette width (`asw`) for `pam` clustering results with true labels and for all random labelings. Generate a boxplot of `asw` values obtained from random labelings. Compare it with the `asw` value obtained from true labels.

```
> d.s <- dist(t(dat.s))
> cl.true <- as.numeric(cl == "B")
> asw <- mean(silhouette(cl.true, d.s)[, 3])

> asw.random <- rep(0, 20)
> for (sim in 1:20) {
+     cl.random = sample(cl.true)
+     asw.random[sim] = mean(silhouette(cl.random, d.s)[, 3])
+ }
> symbols(1, asw, circles = 0.01, ylim = c(-0.1, 0.4), inches = F, bg = "red")
> text(1.2, asw, "Average silhouette value\n for true labels")
> boxplot(data.frame(asw.random), col = "blue", add = T)
```

You will see: The average silhouette value of the `pam` clustering result lies well outside the range of values achieved by random clusters.

6.6. Clustering of genes

Next we will apply hierarchical clustering to the 100 top variance genes. For a easier interpretation of the result one should use the gene names (Symbols) instead of the corresponding Affymetrix identifier. You can get the gene names from the annotation package `hgu95av2.db`.

```
> library(hgu95av2.db)
> gene.names <- as.list(hgu95av2SYMBOL)[rownames(dat)]
> gene.names <- unlist(gene.names[genes.var.select])
> d.g = dist(dat.s)
> hc = hclust(d.g, method = "complete")
> plot(hc, labels = gene.names)
```

Now plot the average silhouette widths for `k=2:20` clusters using `pam` and have a look at the silhouette plot for `k=2`. What do you think: Did we find a structure in the genes?

```
> asw = numeric()
> for (k in 2:20) {
+     asw[k] = pam(dat.s, k)$silinfo$avg.width
+ }
> plot(1:20, asw, xlab = "# clusters", ylab = "average silhouette width", type = "b")
> plot(silhouette(pam(dat.s, 2)))
```

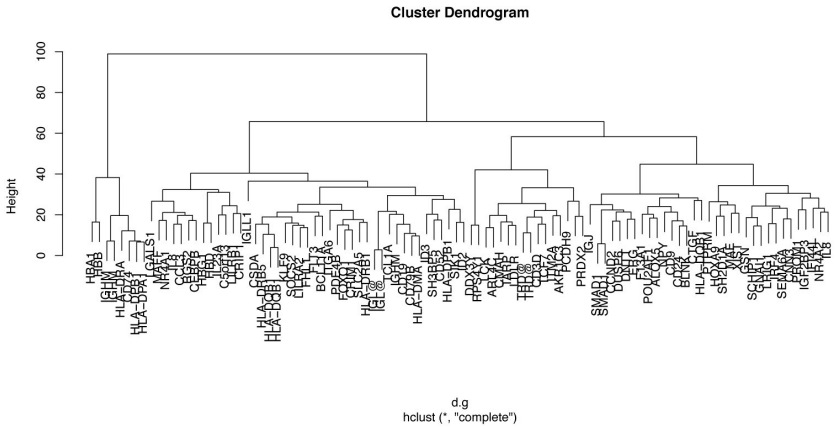


Fig. 6.6: Hierarchical clustering of 100 top variance genes

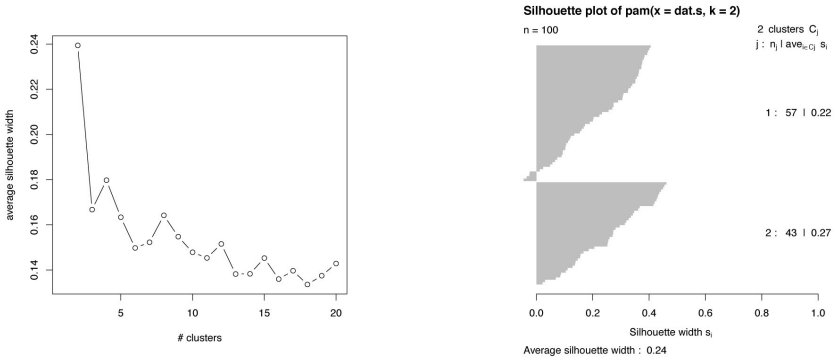


Fig. 6.7: Left: Average silhouette width of the 100 top-variance genes for `k=2:20` clusters. Right: Silhouette plot for `k=2`.

6.7. Presenting results

In the last section we saw that a substructure in the genes is not easy to find. When interpreting heatmap plots like the following, it is important to keep in mind: Any clustering method will eventually produce a clustering; before submitting a manuscript to Nature you should check the validity and stability of the groups you found.

```
> help(heatmap)
> heatmap(dat.s)
```

6.8. How to fake clusterings

For publication of your results it is advantageous to show nice looking cluster plots supporting your claims. Here we demonstrate how to produce almost *any clustering you like* in a

```
> help(heatmap)
> heatmap(dat.s)
```

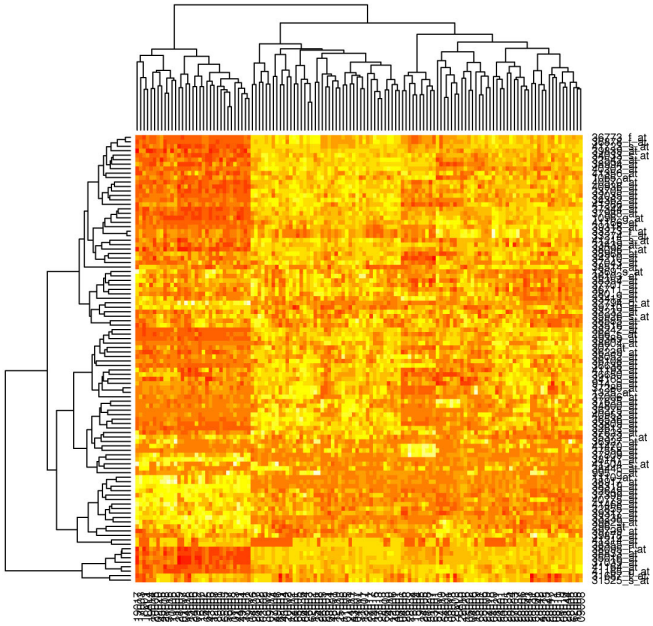


Fig. 6.8: Heatmap representation of all samples and 100 selected high-variance genes. Rows and columns are hierarchically clustered.

situation with few samples but many genes.

First reduce the data set to the first 5 B-cell and T-cell samples. Then permute class labels randomly. The permuted labels do not correspond to biological entities. Nevertheless we will demonstrate how to get a perfect clustering result.

The strategy is easy: Just select 100 genes according to differential expression between the groups defined by the random labels, then perform hierarchical clustering of samples with these genes only.

```
> select = c(which(cl == "B")[1:5], which(cl == "T")[1:5])
> dat.small = dat[, select]
> cl.random = sample(cl[select])
> library(multtest)
> tstat = mt.teststat(dat.small, classlabel = (cl.random == "B"), test = "t")
> genes = order(abs(tstat), decreasing = T)[1:100]
> dat.split = dat.small[genes, ]
> d = dist(t(dat.split))
> hc = hclust(d, method = "complete")
```

```
> par(mfrow = c(1, 2))
> plot(hc, labels = cl.random, main = "Labels used for gene selection", xlab = "")
> plot(hc, labels = cl[select], main = "True labels", xlab = "")
```

Repeat this experiment a few times and compare the two plots. You should also have a look at the heatmap representation of `dat.split`.

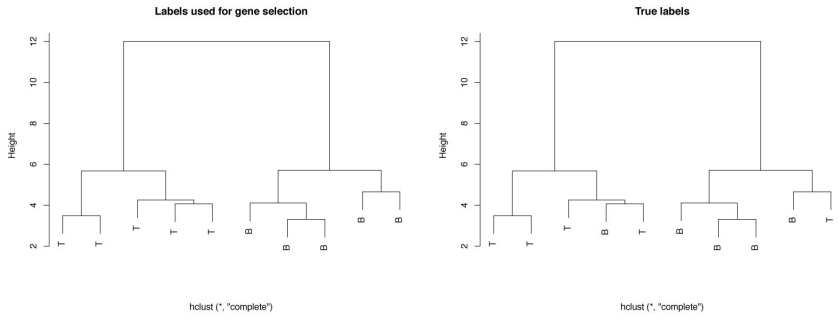


Fig. 6.9: Selecting only highly differential genes leads to well separated groups of samples — even if these groups are defined by random assignments of labels without any biological meaning.

6.9. Ranking is no clustering!

A typical situation in microarray analysis is the following. You are interested in one specific gene and want to identify genes with a similar function. A reasonable solution to this problem is to calculate the distance of all other genes to the gene of interest and then analyze the list of the closest (top-ranking) genes.

We choose the fifth gene out of our 100 high variance genes and print the distances and the names of the 10 genes that are closest to it.

```
> list.distances = as.matrix(d.g)[5, ]
> o = order(list.distances)
> list.distances[o][1:10]
> gene.names[o][1:10]
```

You can see that the original gene of interest is ranked first with a distance 0 to itself, and the second and the seventh gene are replicates of this gene. This is a plausible result.

Another popular approach in this situation is to first cluster all genes and then analyze the genes that are in the same cluster as the original gene of interest. In general this strategy is not suitable since even two objects that are very close to each other can belong to two different clusters if they are close to a cluster boundary line.

In our example, look at the result of the hierarchical clustering of genes obtained above. Define three clusters based on this result and print the clusters of the 10 closest genes identified above.

```
> hc = hclust(d.g, method = "complete")
> cutree(hc, 3)[o][1:10]
```

To which clusters do the original gene of interest and the replicates belong? You can see that even on this low clustering level the replicates are separated. Thus remember: If you are interested in detecting genes that are similar to a gene of your choice, you should always prefer a ranked list to a clustering approach.

6.10. Summary

Scan through this paper once again and identify in each section the main message you have learned. Some of the most important points are:

- You learned about hierarchical clustering, k-means, partitioning around medoids, and silhouette scores.
- Selecting high-variance genes before clustering improves the result.
- Selecting only highly differential genes leads to well separated groups of samples — even if these groups are defined by random assignments of labels without any biological meaning.

If you have any comments, criticisms or suggestions on our lectures, please contact us:
alexa@mpi-inf.mpg.de

6.11. Acknowledgements

Many thanks to Jörg Rahnenführer and Florian Markowetz for sharing large parts of this document.

7

Tutorial Cytoscape

Fidel Ramírez

En este tutorial se va a introducir la plataforma para análisis de redes *Cytoscape* usando como ejemplo una red de interacción de levadura relacionada con la adquisición y metabolismo de la galactosa. Este ejemplo se utilizará para ir introduciendo diversas funcionalidades y conceptos. El orden de este tutorial es importante ya que la mayoría de las secciones usan los resultados obtenidos en secciones precedentes. Por ello, en caso de tener que abandonar el tutorial lo mejor es guardar la sesión en que esté trabajando.

7.1. ¿Qué es Cytoscape?

Cytoscape (Shannon *et al.*, 2003; Killcoyne *et al.*, 2009) es una herramienta para la visualización de redes de interacción molecular y rutas metabólicas que además permite la integración de perfiles de expresión y otros datos moleculares. Cytoscape es un software de bioinformática creado gracias a la colaboración derivada del modelo de código libre. Actualmente su desarrollo, tanto del programa en si, como de sus extensiones (plugins) es bastante activo. Cytoscape apareció por primera vez en el año 2002 con el lanzamiento de la versión 0.9 por parte del Institute of Systems Biology de Seatle. Hoy en día Cytoscape va en la versión 2.6 y su desarrollado está a cargo de un consorcio internacional. También, un creciente número de instituciones han contribuido con la creación de nuevas extensiones para Cytoscape que amplían enormemente su funcionalidad. Si bien, Cytoscape está enfocado para la comunidad biológica igualmente puede ser usado para visualizar otros tipos de redes, por ejemplo redes sociales.

7.2. Estructura de Cytoscape

En esta sección se introduce la interfaz de usuario de Cytoscape. En primer lugar se dará un vistazo general que será luego complementado con la descripción de los menús y de los plugins que extienden su funcionalidad.

7.2.1. Interfaz de usuario

Al iniciar Cytoscape debe aparecer una ventana como en la Figura 7.1

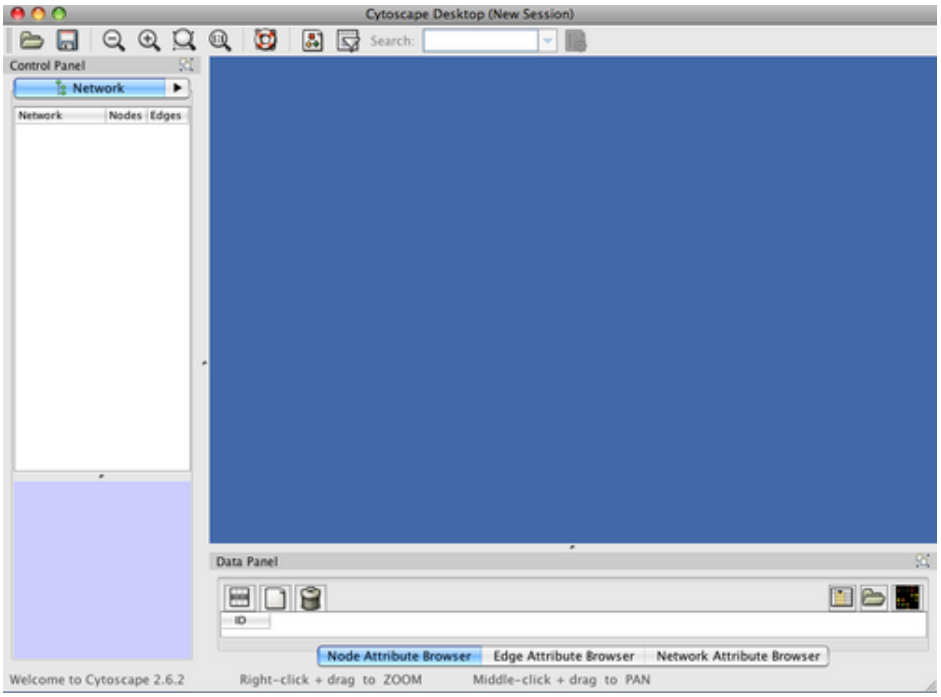


Fig. 7.1: Ventana principal de Cytoscape. Las secciones principales son: La barra de herramientas en la parte superior, el panel de control a la derecha, la vista principal a la izquierda y el panel de datos en la parte inferior.

- En la parte superior de la ventana de Cytoscape aparece la barra de herramientas. La función de cada uno de los botones se muestra al llevar el puntero del ratón sobre ellos.
- En la parte superior derecha está la ventana de vista principal dónde las redes pueden ser visualizadas. Esta región está inicialmente en blanco.
- A la izquierda está el panel de control (Network Management panel). Éste lista las redes disponibles por nombre y proporciona información sobre el número de nodos y de aristas.
- Bajo el panel de control se encuentra el panel de la sinopsis de la red (Network overview panel). Este panel se usa para desplazarse por la red cuando esta no cabe totalmente en la pantalla. También se puede navegar una red usando el botón central del ratón desde la vista principal.
- En la parte inferior derecha está el panel de datos que puede ser usado para visualizar atributos de los nodos, las aristas o de las redes.

Tanto el panel de control como el de datos pueden contener pestañas que reciben el nombre de Cytopaneles. Cualquiera de estos paneles puede ser desacoplado haciendo clic en el pequeño icono en la parte superior derecha del Cytopanel.

7.3. Visualización de redes

Si bien, en Cytoscape se pueden crear redes nodo por nodo usando el panel de edición localizado en una de las pestañas del panel de control, lo más sencillo es comenzar trabajando con una red ya existente que hace parte de la carpeta de ejemplos con que viene Cytoscape.

7.3.1. Cargar una red sencilla

- Vaya a **File** → **Import** → **Network (multiple file types)**.
- Debe aparecer el cuadro de dialogo "Import Network File".
- Seleccione **Local** y continúe.
- Abra la carpeta **sampleData** y seleccione el archivo `galFiltered.sif`. Haga click en Abrir y luego en **Import**.

Un cuadro de diálogo como el que se muestra en la Figura 7.2 debe aparecer en la pantalla. Allí se le informa el número de nodos y de aristas encontrado. En el panel de control, ahora aparece también el nombre de la red. Como aún no se le ha dado formato a la visualización de la red, esta se ve como un cuadrado lleno de puntos y rayas.

El archivo con formato `.sif`[1] que acaba de cargar es un simple archivo de texto que contiene tres columnas: nodo fuente, tipo de interacción y el o los nodos destino. Usualmente el nodo fuente y el nodo destino son identificadores o nombres de genes/proteínas usados para definir los nodos. El tipo de interacción se usa como etiqueta para las aristas. Intente abrir el archivo **galFiltered.sif** usando el *notepad* en windows o `less` en Unix para reconocer una estructura similar a la que se muestra a continuación:

```
nodoA <tipo de interacción> nodoB
nodoC <tipo de interacción> nodoA
nodoD <tipo de interacción> nodoE nodoF nodoB
nodoG
...
nodoY <tipo de interacción> nodoZ
```

Un ejemplo concreto sería:

```
P53  pp MDM2
P53  pp USP7
MDM2 pp PML
```

[1] Para mayor información visitar http://www.cytoscape.org/cgi-bin/moin.cgi/Cytoscape_User_Manual/Network_Formats

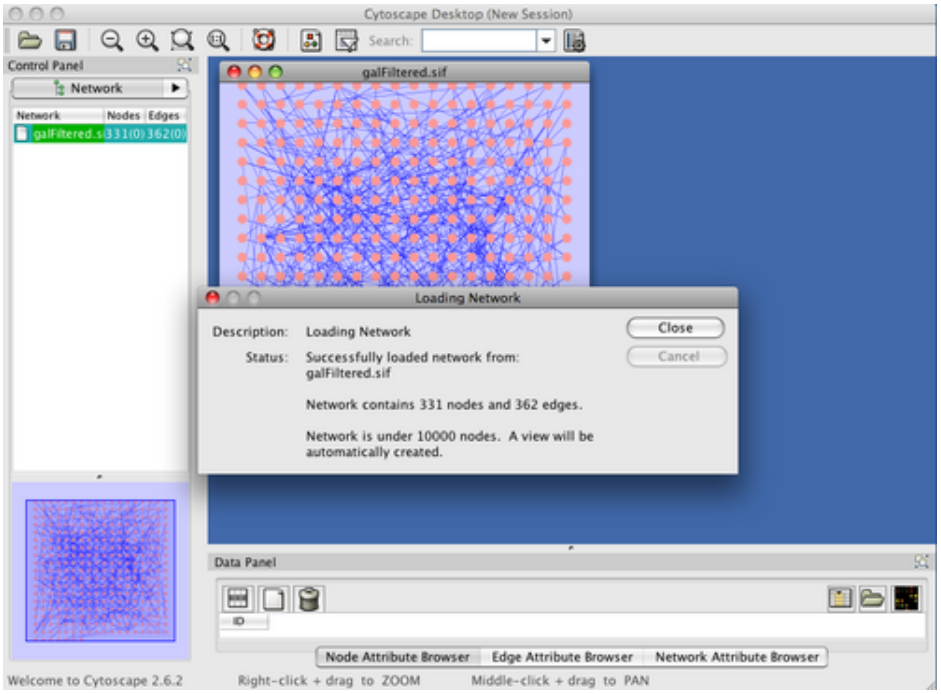


Fig. 7.2: Mensaje de finalización del cargado de una nueva red en Cytoscape.

En biología de sistemas `pp` y `pd` son tipos de interacción comunes para interacciónes proteína-proteína y proteína-DNA respectivamente.

Aparte del formato .sif, Cytoscape también incluye una útil manera de importar archivos en formato MS Excel o de texto delimitado. Esto se encuentra en el menú **File → Import → Network from Table (Text/MS Excel)**.

El archivo `galFiltered.sif` proviene del artículo de IDEKER *et al.* (2001). En este trabajo de investigación se estudió el uso de galactosa por parte de levadura usando datos de interacción de proteínas y datos de expresión de mRNA y de proteínas después de perturbar un número de genes seleccionados. En la carpeta **sampleData** encontrará otras versiones del mismo archivo en diferentes formatos así como datos adicionales.

7.3.2. Darle formato a la visualización de la red

Para mejorar la visualización de la red haga lo siguiente:

- Vaya a **Layout → yFiles → Organic**.
- Después de unos segundos verá la nueva visualización.

Experimente con diferentes tipos de formato como 'yFiles → circular' o 'Cytoscape Layouts → Spring Embedded'. Cada disposición (layout) es computada usando algoritmos diferentes

que resaltan algunas características de la red. Por ejemplo las disposiciones 'organic' y 'spring embedded' hacen énfasis en posibles grupos de nodos.

Usando el ratón se pueden seleccionar nodos individuales o grupos de nodos. Esta selección se puede arrastrar para cambiarla de posición.

7.4. Análisis de la topología de la red usando Network Analyzer

NetworkAnalyzer (Assenov *et al.*, 2008) es una extensión para Cytoscape que computa más de veinte parámetros distintos incluyendo *coeficiente de agrupamiento (clustering coefficient)*, *excentricidad*, *diámetro* y *radio* de una red entre otros. Usando NetworkAnalyzer es posible comenzar a detectar y visualizar nodos de interés biológico, por ejemplo aquellos con más conexiones a otros nodos puden indicar un rol regulador o, en caso de una red experimental obtenida mediante yeast two-hybrid, también podrían estar indicando problemas experimentales, como la presencia de genes autoactivadores.

- Para comenzar a utilizar NetworkAnalyzer vaya a **Plugins** → **Network Analysis** → **Analyze Network**.
- Seleccione **Treat this network as undirected**.
- Debe aparecer una ventana como la que se muestra en la Figura 7.3. Fíjese en los diferentes parámetros globales que allí se muestran. ¿Qué significa que en la red haya 26 componentes conectados?
- Fíjese en el valor para **Characteristic path length**. Este es el promedio del tamaño de las rutas más cortas (*shortest paths*) entre dos nodos. ¿Qué interpretación se le puede dar a este resultado? ¿Será una red *Mundo pequeño* (small world)?
- El coeficiente de agrupamiento refleja la interconexión existente entre los vecinos de un nodo. Un coeficiente de agrupamiento cercano a 1 indica la presencia de muchos grupos de nodos interconectados entre si. ¿Cómo interpreta usted el coeficiente de agrupamiento de la red de uso de galactosa?
- El parámetro **Network heterogeneity** refleja la tendencia de una red a contener concentradores (*hubs*) es decir nodos con muchas conexiones.
- La densidad de la red es 0 cuando ningún nodo está conectado a otro y es 1 cuando todos los nodos están conectados con todos. Para esta red la densidad es 0.007 lo cual quiere decir que los nodos casi no estám conectados los unos con los otros.

Las diferentes pestañas de NetworkAnalyzer permiten ver el resultado de los parámetros de la red. Por ejemplo, haga clic en la pestaña **Node Degree Distribution** para ver, en un gráfico con escala doble logarítmica, la distribución del los grados de la red. Fíjese que hay muchos nodos con pocas aristas (esquina superior izquierda) y pocos nodos con muchas aristas

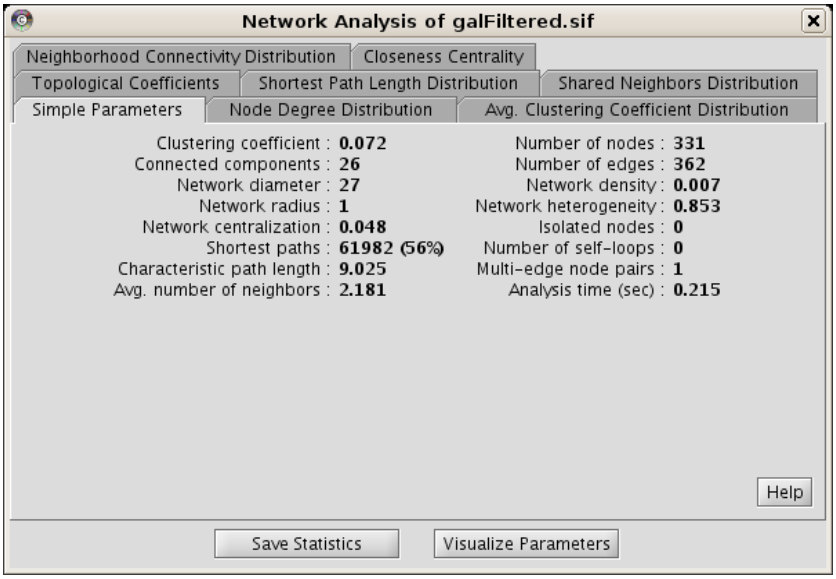


Fig. 7.3: Ventana de resultados de la extensión NetworkAnalyzer donde se observan los parámetros topológicos globales de la red de uso de galactosa.

(esquina inferior derecha). Este tipo de distribución es muy común en varios tipos de redes incluyendo las biológicas. Si usted está trabajando con una red biológica y no presenta esta distribución, llamada *libre de escala (scale free)*, procure revisar que no haya cometido algún error. Al seleccionar **Fit Power Law** puede ver una ventana como la que se muestra en la Figura 7.4 donde han ajustado los valores de la función $\ln y = \alpha \ln x + \ln \beta$ usando el método de los mínimos cuadrados para dibujar la linea[2].

NetworkAnalyzer agrega los parámetros que computa a los atributos de los nodos lo cual now permite visualizarlos en Cytoscape. Sin embargo, existe una manera fácil de ver estos parámetros usando una funcionalidad incluida dentro de NetworkAnalyzer.

- Cierre el cuadro de diálogo de NetworkAnalyzer. No se preocupe por guardar los datos en este caso. La opción de guardar es especialmente útil cuando se tiene una red grande para la cual el cómputo de los parámetros topológicos puede tardar mucho.
- Vaya a **Plugins → Network Analysis → Visualize Computed Parameters**.
- Seleccione: **Map node size to**: Degree.
- Seleccione: **Map node color to**: Eccentricity.
- La visualización de la red debe ser similar a la Figura 7.5. Fácilmente puede encontrar los grupos de genes con un mayor número de interacciones y más centrales de la red. En la

[2]Una mayor información sobre los diferentes parámetros topológicos se encuentra en el manual en linea de NetworkAnalyzer http://med.bioinf.mpi-inf.mpg.de/netanalyzer/help/2.6.1/index.html.

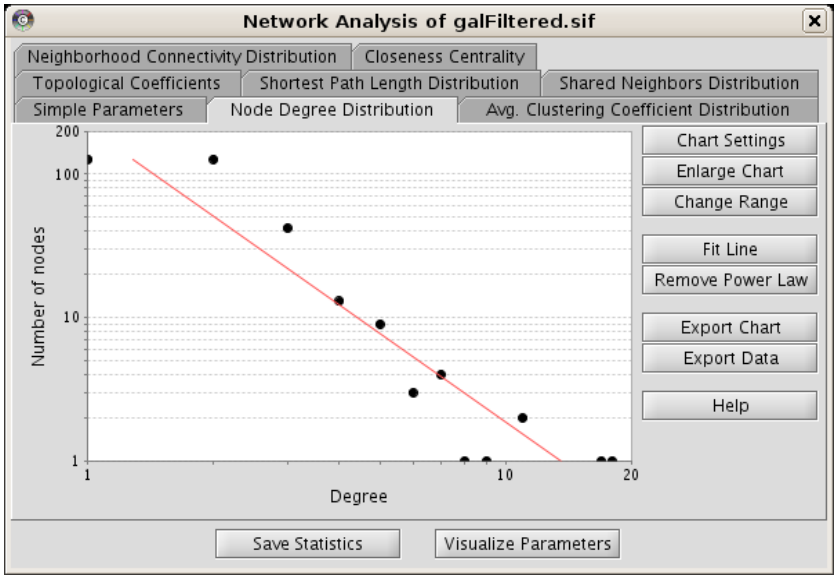


Fig. 7.4: Distribución del grado de los nodos en la red. Éstos siguen una distribución *libre de escala (scale free)*, llamada así porque se puede representar mediante una función exponencial (power law) que no cambia su representación gráfica aun cuando los parámetros de la función sean multiplicados por otro valor.

sección 7.4.2 se cargarán atributos extra que permitirán reconocer mejor los genes que se han resaltado. Por ahora solo podemos observar su identificador.

- Seleccione diferentes parámetros y compare los resultados.

7.4.1. Uso de filtros

Usando NetworkAnalyzer y la opción de filtros de Cytoscape se pueden generar nuevas visualizaciones, por ejemplo para resaltar factores de transcripción y la relación entre ellos. Para ello, en esta sección vamos a seleccionar solo las interacciones proteína–DNA de la red mediante la creación de un filtro. A partir de esta selcción vamos a crear una nueva visualización.

- Diríjase a la pestaña **Filters** de Cytoscape, en el panel de control.
- Haga clic en el botón **Option** y seleccione **Create new filter**. Elija un nombre para el nuevo filtro. Por ejemplo "protein–dna".
- En el panel **Filter Definition** seleccione en el menú desplegable **Attribute/Filter** la opción **edge.interaction** y haga clic en el botón **Add.**
- En el nuevo menú desplegable seleccione la opción **pd** para que el filtro solo seleccione las interacciones proteína–DNA. NOTA: El archivo `.sif` que usted cargó al inicio contenía dos tipos de interacciones **pp** (proteína–proteína) y **pd** (proteína–DNA).

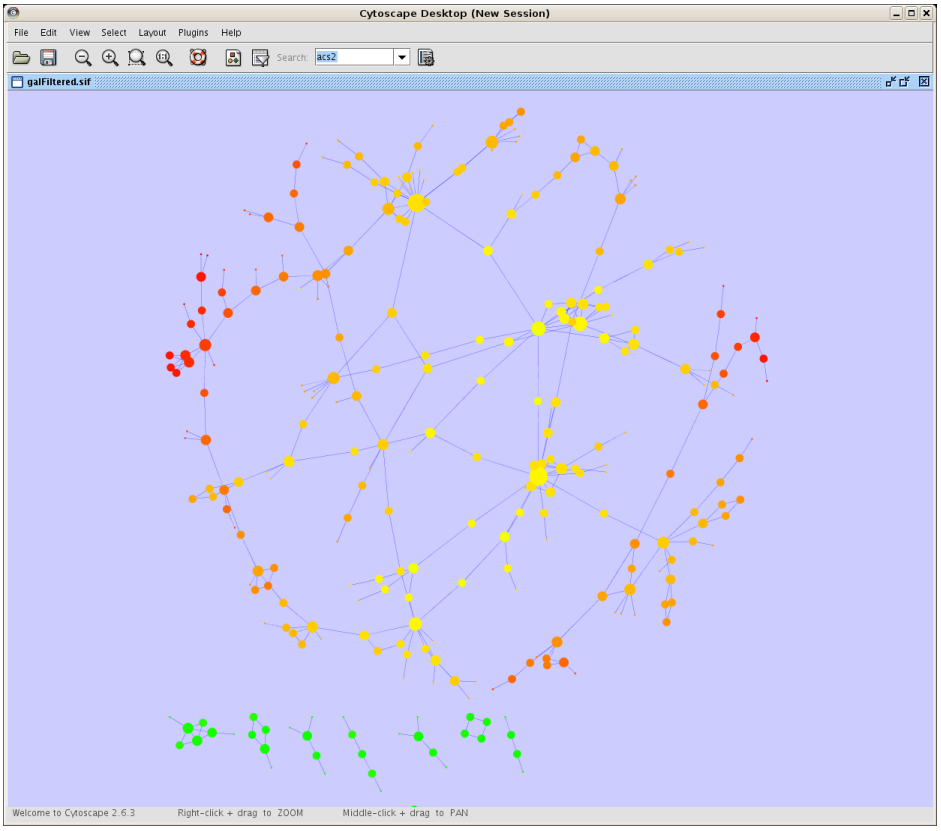


Fig. 7.5: Visualización de la red donde el tamaño de los nodos es proporcional a su grado y donde su color es más oscuro o más claro dependiendo de si su excentricidad es más alta o baja respectivamente.

- Haga clic en el botón **Apply.**
- Ahora deben aparecer seleccionadas (en rojo) todas las interacciones proteína–DNA presentes en la red. Sin embargo, lo más útil es crear una nueva vista de la red solo con los nodos y aristas selccionados. Para ello vaya a **File** → **New** → **Network** → **From selected nodes, selected edges**.
- Aparecerá en la pantalla una nueva visualización de la red que solo contiene las interacciones seleccionadas. En el panel de control puede comprobar que la red original continúa allí.
- Utilice de nuevo la extensión NetworkAnalyzer para computar los nuevos parámetros de la red. Visualice el grado de la red como el tamaño de los nodos.
- Ahora vaya al menú **Layout** → **Cytoscape Layouts** → **Hierarchical Layout.**
- Vaya a **Layout** → **Rotate**. Aparecerá un nuevo panel en la esquina inferior izquierda. Gire la figura 180 grados.

- Debe obtener un resultado similar a la Figura 7.6. Más adelante agregaremos las etiquetas adecuadas para los genes.

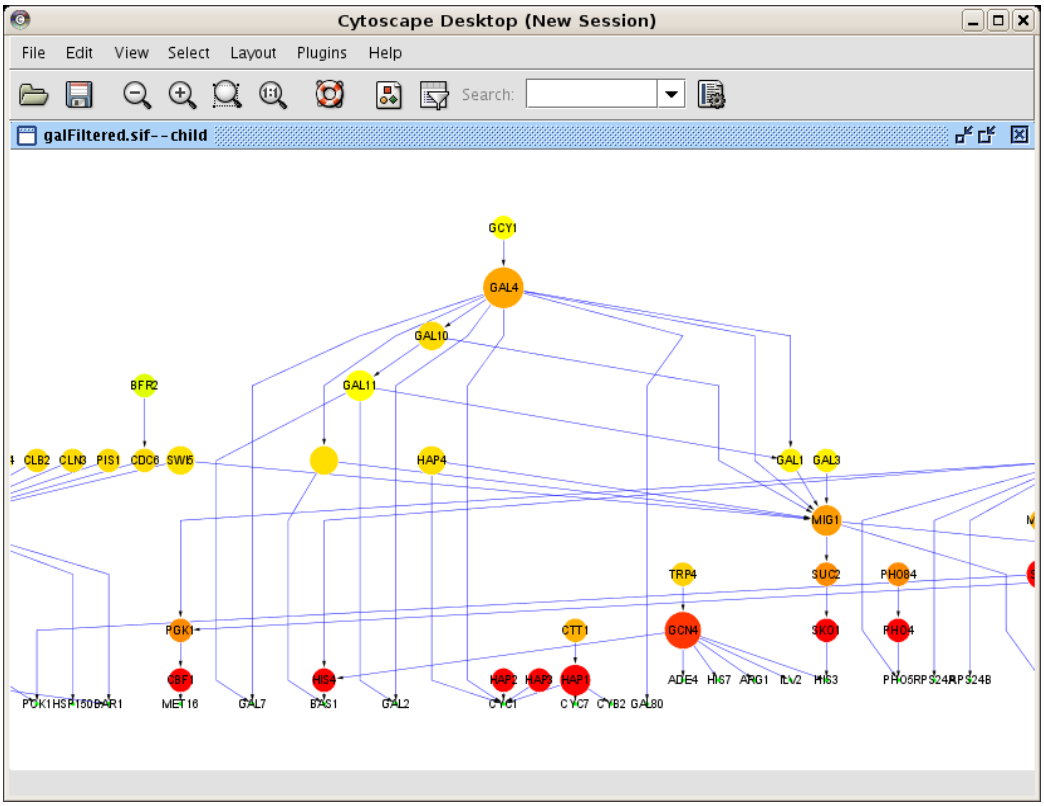


Fig. 7.6: Visualización de las interacciones proteína–DNA de la red usando la disposición jerárquica (Hierarchical layout). El tamaño de los nodos es proporcional al número de genes controlado por ellos.

7.4.2. Importar archivos de atributos

Aparte de cargar redes en Cytoscape, también se pueden cargar archivos de atributos tanto para nodos como para aristas. En el siguiente ejemplo vamos a cargar los valores de cambio de expresión disponibles para tres deleciones: *gal1*, *gal4* y *gal80*.

- Vaya a **File → Import → Attribute from Table (Text/MS Excel).**
- Debe aparecer el cuadro de diálogo **Import Attribute from Table.**
- Verifique que seleccionó la opción adecuada para importar atributos y no para importar redes puesto que los nombres en el menú son parecidos.
- Haga click en el botón **Select File(s)** y a continuación seleccione el archivo **galExpData.pvals** de la carpeta **sampleData** y ábralo.

- En la sección **Advanced** , seleccione **Show Text File Import Options.**
- Por defecto el separador **Tab** aparece seleccionado. Seleccione el separador **Space.**
- En la sección de **Attribute Names** seleccione **Transfer first line attribute names**. Verá como las columnas de la parte inferior adquieren los nuevos nombres.
- También en la sección **Advanced**, seleccione **Import everything** para indicarle a Cytoscape que cargue todos los datos y no solo ellos que coincidan con las redes ya cargadas.

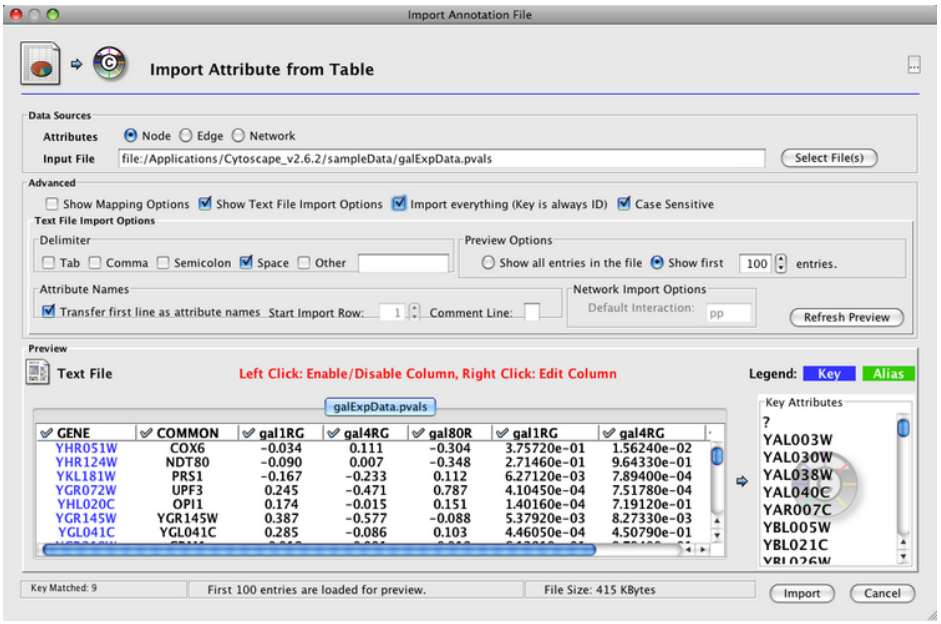


Fig. 7.7: Cuadro de dialogo para importar atributos de nodos a partir de un archivo de texto o de MS Excel.

Si hiciera clic en **Import**, vería un mensaje quejándose por los nombres de algunos atributos duplicados. Observe que las columnas 6, 7 y 8 tienen el mismo nombre que las columnas 3, 4 y 5. Los pasos siguientes muestran cómo cambiar los nombres de las columnas, sin tener que empezar de nuevo desde MS Excel.

- Haga clic en el encabezado de la columna con el primer nombre duplicado (columna 6, gal1RG) para abrir el cuadro de diálogo **Set Attribute Name and Type.**
- Agregue el sufijo 'pval' al nombre de la columna (por ejemplo, 'gal1RGpval') para distinguir la columna que contiene los valores p.
- Repita estos últimos dos pasos con las columnas 7 y 8 (gal4RG y gal80R).
- Haga clic en **Import**

Ahora a la red de uso de galactosa se le han agregado los datos de expresión de los diferentes genes para cada una de las deleciones así como los valores p. También se ha agregado un atributo que contiene el símbolo de los genes. Para verificar que los datos han sido cargados debe ir al **panel de datos** en la parte inferior de la ventana de Cytoscape.

- Busque el botón **Select Attributes**  en el **panel de datos**
- Seleccione los atributos: **gal1RG**, **gal4RG**, y **gal80R**
- Regrese al **panel de datos** haciendo clic por fuera de la lista de atributos
- Con el mouse, seleccione algunos nodos de la red (ctrl-A selecciona todos los nodos). Los datos de expresión que usted haya seleccionado aparecerán ahora en el **panel de datos**.

Más adelante, en la sección 7.7, se hablará de como importar atributos usando servicios en linea.

7.5. Visualización de datos con VizMapper

La manera como se configura la visualización de los datos en Cytoscape es utilizando el VizMapper. Éste se encuentra en una de las pestañas del **panel de control**. Usando el VizMapper se pueden usar los valores de los atributos para cambiar las propiedades tanto de los nodos como de las aristas. Si aún no ha cargado los datos de cambio de expresión para la red de levadura revise por favor la sección anterior antes de continuar.

7.5.1. Cambio de la etiqueta de los nodos

- Localice la pestaña **VizMapper** en el panel de control o acceda a el a través del botón en la barra de herramientas  .
- Encontrará un menú desplegable en la sección **Current Visual Style**. Seleccione diferentes opciones y observe el efecto en la imagen de la red.
- También encontrará una serie de opciones disponibles para cambiar diversos aspectos de la red. Por ejemplo el tipo de línea de las aristas, el color del borde de los nodos etc.
- Comience por seleccionar el estilo visual **default** y por hacer clic en el botón de opciones 
- Seleccione **Copy existing Visual Style** y escoja un nombre para su nuevo estilo.
- Ahora localice **Node Label** y haga clic donde dice **ID**. Aparecerá un menú desplegable que contiene los nombres de todos los atributos de nodo disponibles.

- Seleccione el atributo **COMMON**. Ahora los nodos aparecerán marcados con el símbolo del gen y no con el identificador. Es posible que tenga que hacer zoom en la imagen para poder ver los cambios. NOTA: El atributo que contiene los símbolos de los genes fue cargado previamente junto con los valores de expresión.

- Ahora la visualización de la red debe ser más parecida a la de la Figura 7.6. ¿Qué se puede opinar sobre el papel de *gal4* en la red?

7.5.2. Visualizar datos de expresión

En este ejemplo vamos a cambiar el color de los nodos en relación con su expresión.

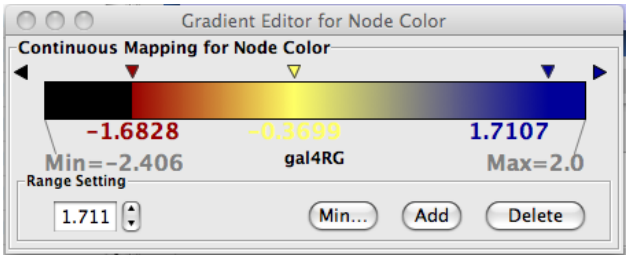


Fig. 7.8: Cuadro de selección de color para valores continuos usando VizMapper.

- Ahora haga clic en **Node Color** y a continuación haga doble clic para cambiar las opciones.

- Busque de nuevo **Node Color** al comienzo de la lista y haga clic en él de nuevo. Seleccione el atributo *gal4RG*.

- Ahora seleccione el tipo de mapeo continuo. Esto quiere decir que vamos a asignar el color del nodo a un atributo de tipo continuo. Otros valores son "Passthrough mapper" y "Discrete".

- Haga clic en **Graphical view**. Debe aparecer un cuadro como el que se muestra en la Figura 7.8.

- Haga clic en el botón **Min...** y seleccione los valores mínimos y máximos -2 y 2 respectivamente.

- Haciendo clic en los triángulos puede seleccionar los colores para diferentes valores. Si desea más colores use el botón **Add**.

- Cierre el cuadro de selección de color.

Ahora, en la visualización de la red, los genes cuya expresión es menor comparada con la referencia silvestre deben tener un color rojo. Aquellos genes que no cambiaron su expresión deben aparecer amarillos y los que aumentaron su expresión deben verse azules (suponiendo

que la selección de colores sea como la de la Figura 7.8). Como el atributo seleccionado para establecer el color de los nodos es el que corresponde a los valores de expresión de los genes ante la deleción de *gal4*, debe observar que los genes controlados por este gen disminuyen su expresión.

Usando el VizMapper cambie el atributo que originalmente seleccionó para el color de nodo, de *gal4RG* a *gal80R*. ¿Cómo puede interpretar estos cambios de expresión? Fíjese en la Figura 7.9 muestra el modelo del uso de galactosa. ¿Piensa usted que los cambios de expresión que se observan con las diferentes visualizaciones soportan el modelo?

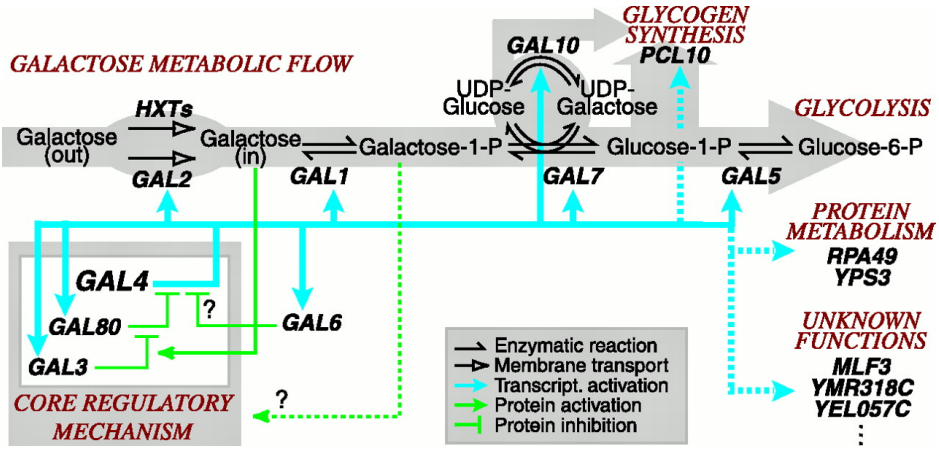


Fig. 7.9: Modelo del uso de galactosa en lavadura. *GAL4, GAL80* y *GAL3* son los principales reguladores del metabolismo de galactosa. Tomado de Ideker *et al.* (2001)

Como habrá podido observar, la manipulación de la visualización de la red es un medio de ayuda importante para aclarar o establecer nuevas hipótesis de estudio. Con la ayuda de Cytoscape se pueden crear y comparar múltiples visualizaciones que reflejen diferentes características de la red.

Puesto que algunas personalizaciones llegan a ser complejas, es posible importar y exportar archivos de propiedades para VizMapper. Encontrará estas opciones en el menú de **File**. Igualmente si guarda su sesión, la personalización que halla hecho también será guardada. Mediante la personalización de varias opciones se puede obtener una red como la de la Figura 7.10. Termine esta sección guardando la sesión (**File** → **Save**).

7.6. Función de los genes en la red

Una parte importante del análisis de una red es la determinación de las funciones, rutas metabólicas y lugares de expresión de los genes dentro de la célula. Para ello, Cytoscape cuenta con varias utilidades que asisten al usuario en este proceso y que involucran la conexión a diversos servidores que pueden contener la información requerida.

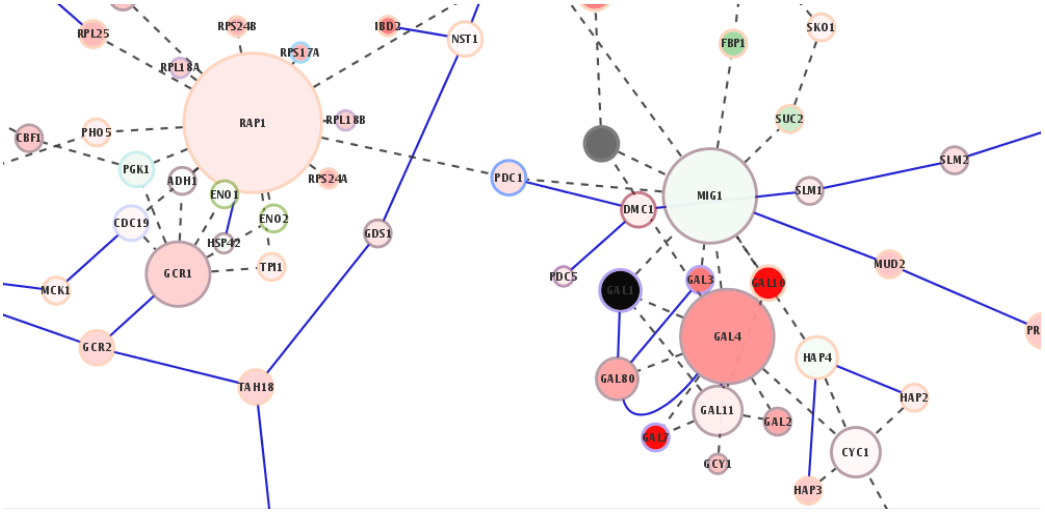


Fig. 7.10: Usando VizMapper muchas de las propiedades de los nodos y las aristas se pueden personalizar incluyendo color del nodo y de las aristas, tamaño etc. La visualización que se muestra se puede cargar en Cytoscape usando el archivo **galFiltered.cys** en la carpeta **sampleData**.

7.6.1. Importar ontologías y asociaciones

- Vaya a **File** *rightarrow* **Import** *rightarrow* **Ontology an Annotations**.
- Seleccione **Gene Association file for Saccharomyces cerevisiae** en el primer menú desplegable de la ventana (Fig. 7.11

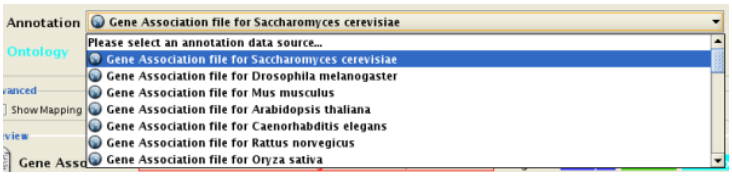


Fig. 7.11: Selección del tipo de asociación requerida. En este caso el respectivo para *Saccharomyces cerevisiae*.

- Ahora seleccione **Yeast GO slim** en el siguiente menú desplegable. Las anotaciones GO slim sólo incluyen conceptos generales con respecto a la estructura de GO. Para mayor información al respecto puede visitar: http://www.geneontology.org/GO.slims.shtml
- Aparecerán en la parte inferior de la ventana varias columnas. Seleccione **Show Mapping Options**.
- Lo que usted está viendo ahora es la manera como los nuevos datos descargados se relacionan con los datos existentes en la red. Por defecto aparecerá que la columna *DB_Object_Symbol*, de los datos descargados, está siendo relacionada al atributo ID de su red. Cambie el atributo de ID a COMMON.

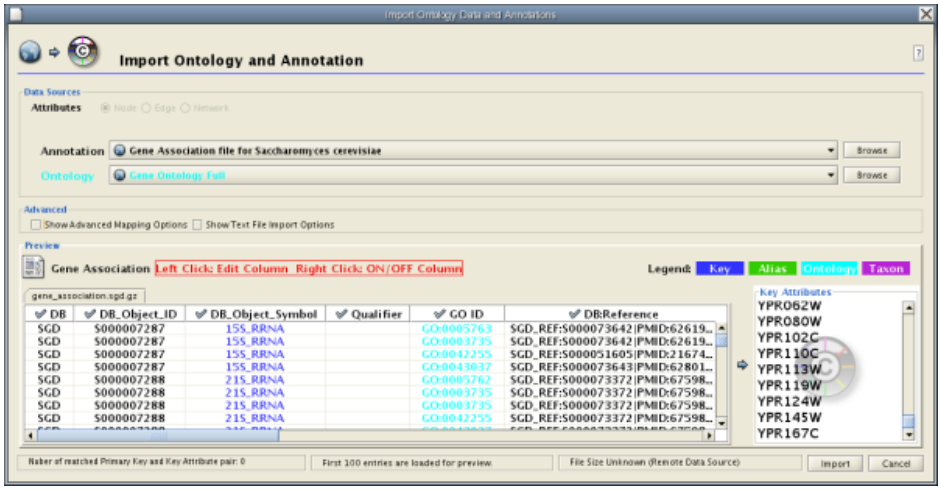


Fig. 7.12: Ventana principal para la adquisición de ontologías y asociaciones.

- Haga clic en **Import**. Verá que aparece una nueva red en el panel de control que por defecto no tiene visualización. Para visualizarla, haga clic-derecho sobre **Yeast GO slim** y seleccione **create view**. Si usted hubiera cargado la ontología completa de GO, lo más recomendable sería no visualizarla por la gran cantidad de nodos que tiene.

- Ahora, nuevos atributos han sido cargados a la red y usted los puede visualizar en el panel de datos. Recuerde que primero tiene que seleccionar los atributos que quiere ver usando el botón de seleccionar atributos.

- Una manera útil de ver los resultados es cambiando la etiqueta de los nodos (como se hizo en la sección anterior con VizMapper) usando ahora alguna de las categorías de GO, por ejemplo *Biological process* (Figura 7.14).

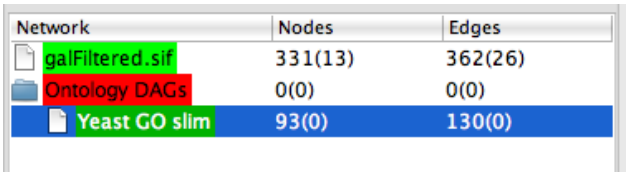


Fig. 7.13: Visualización de una ontología. Después de cargar una ontología y sus respectivas asociaciones aparece una nueva red en el panel de control.

7.6.2. Análisis de funciones sobre-representadas usando BiNGO

BiNGO (Maere *et al.*, 2005) es una útil extensión para Cytoscape que permite el análisis funcional de una red al permitir encontrar aquellas funciones que se encuentren sobre o sub representadas. BiNGO es el acrónimo para: Biological Network Gene Ontology tool. Una gran

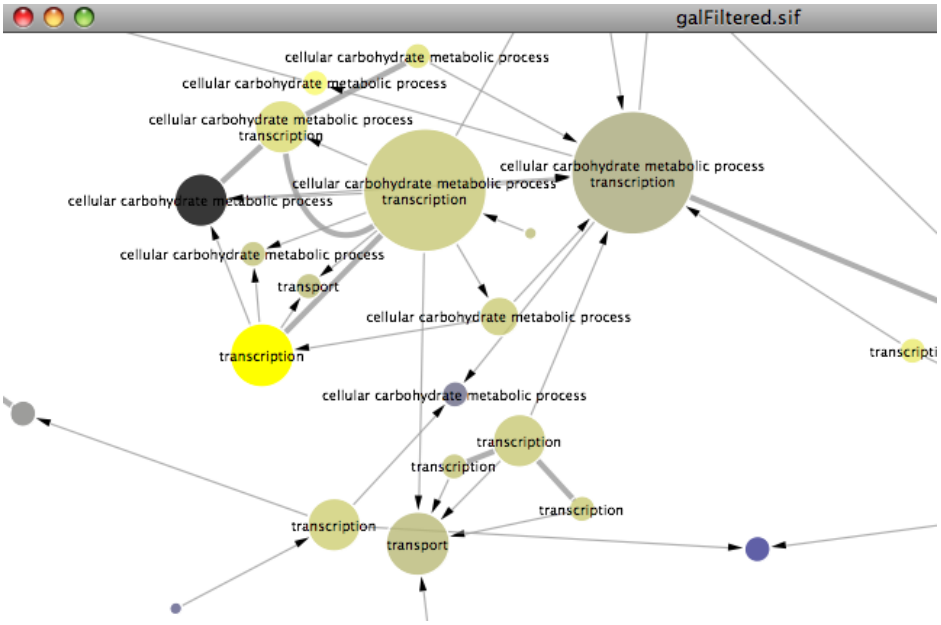


Fig. 7.14: Las anotaciones de GO se pueden visualizar fácilmente cambiando la etiqueta de los nodos. Para esta imagen se uso como atributo: "*annotation.GO BIOLOGICAL PROCESS*."

ventaja de esta extensión es que permite la visualización interactiva de los resultados. En esta sección vamos a hacer el análisis funcional de la red con la cual hemos venido trabajando. Para cargar la red en Cytoscape siga los pasos de la sección 7.3.1.

- Seleccione todos los nodos de la red con Ctrl-a.
- Vaya a **Plugins** → **BiNGO**[3].
- Aparecerá una ventana como la que se muestra en la Figura 7.15.
- Seleccione un nombre donde dice **Cluster name**.
- Asegúrese de que **Get Cluster from Network** esté seleccionado.
- Seleccione las casillas de **Overrepresentation** y **Visualization**.
- Seleccione alguno de los test disponibles. Los autores indican que el test hipergeométrico. es más exacto mientras que el test binomial es más rápido.
- Seleccione una corrección de para múltiples tests. Los autores recomiendan *Benjamini & Hochberg's FDR correction*.
- Seleccione el **significance level**, por ejemplo 0.01.

[3]Si esta opción no aparece, la puede instalar usando yendo a **Plugins** → **Manage Plugins. Lo encontrará en la sección *Functional Enrichment*.**

- Puesto que la idea es visualizar aquellas categorías de GO que están sobre representadas después de la corrección de test múltiples, seleccione visualizar **Overrepresented categories after correction**.

- Escoja como conjunto de referencia todas las anotaciones, en este caso, para el genoma de levadura.[4]

- Seleccione la ontología **GO_Biological_Process** y el organismo **Saccharomyces cerevisiae.**[5]

- Haga clic en el botón **Start BiNGO.**

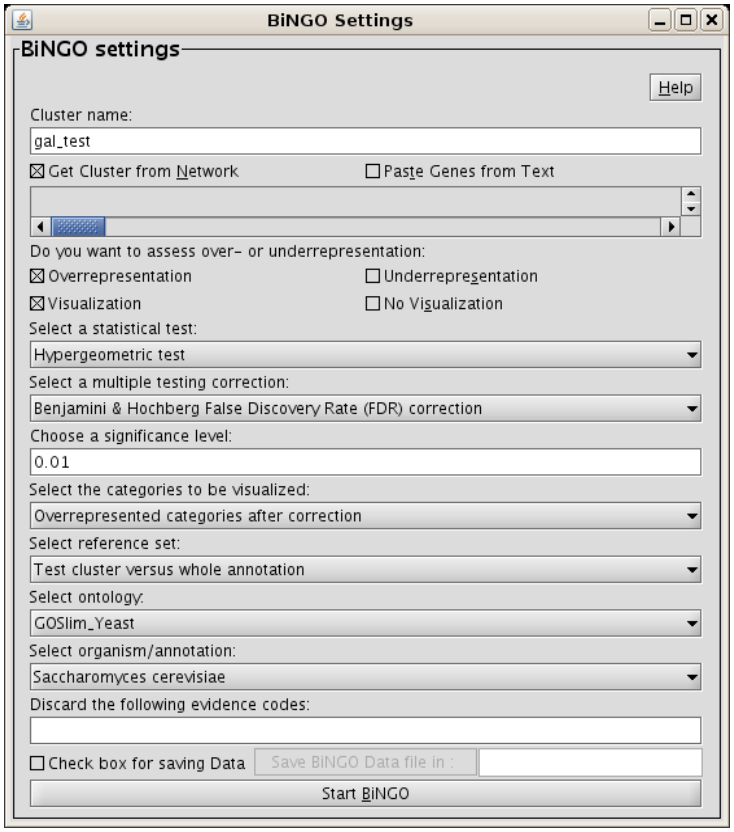


Fig. 7.15: Ventana de configuración de BiNGO para determinar el enriquecimiento de genes en diferentes categorías de GO.

[4]En muchos casos es útil seleccionar una parte de la red y comparar si existe alguna categoría sobre o sub representada con respecto a la totalidad de las anotaciones de la red y no a la totalidad de las anotaciones del genoma del organismo.

[5]Aparte de las otras categorías de GO disponibles, Cellular Compartment y Molecular Function, en ocasiones también es útil usar los *GOSlims* para darse una idea de las categorías generales de la red que aparecen sobre o sub representadas.

BiNGO creará una nueva red que contiene la estructura de de la Ontología seleccionada, en este caso **Biological process**. Los nodos blancos son aquellos que no están sobre representados, pero se muestran porque contienen nodos hijos que si están sobre representados. Los nodos más anaranjados son aquellos con un valor p más pequeño. El tamaño del nodo es proporcional al número de genes en la red de galactosa que contiene esa anotación.

Dependiendo del interés del investigador puede ser más relevante fijarse en los nodos grandes cuya anotación es más general, como por ejemplo *gene expression* o *cellular metabolic process* o por el contrario se puede prestar más atención a los nodos pequeños que involucran menos genes, como por ejemplo **galactose metabolic process** (Figura 7.16)

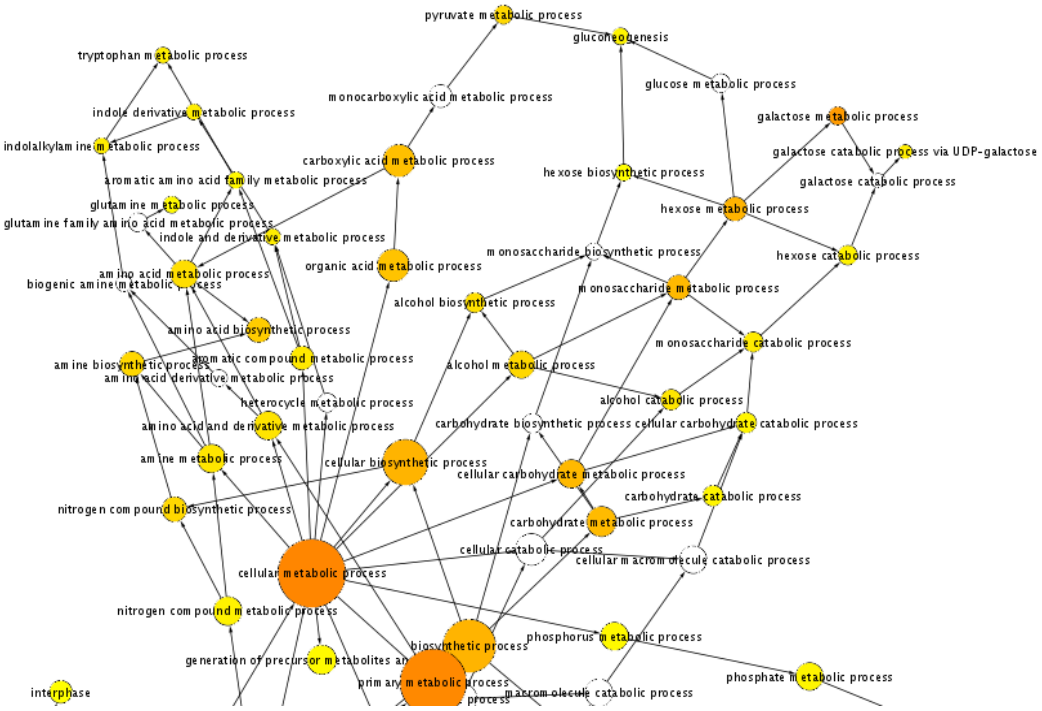


Fig. 7.16: BiNGO genera una nueva visualización que contiene la ontología seleccionada. El tamaño de los nodos es proporcional al número de genes anotados con ese término y el color está relacionado con los valores p de enriquecimiento de genes.

- Navegue la red que contiene la ontología. Fíjese que esta principalmente contiene genes cuya función está relacionada con procesos metabólicos y de regulación (los nodos anaranjados más grandes).

- Los resultados de BiNGO aparecen en la ventana titulada **BiNGO output**. Estos resultados están ordenados por valor p corregido de menor a mayor. Busque la categoría **galactose metabolic process** y marque el respectivo cuadro de selección como se observa en la Figura 7.17.

- Vaya al panel de control y seleccione la red de galactosa (galFiltered.sif)
- Regrese a la ventana de resultados de BiNGO y haga clic en el botón **Select nodes**. En la red de galactosa, aquellos nodos con la anotación que usted escogió aparecerán seleccionados en la red.
- Repita los pasos anteriores para seleccionar otras categorías los genes asociados a ellas. Esto le permite comenzar a conocer mejor las funciones de la red (Figura. 7.18).

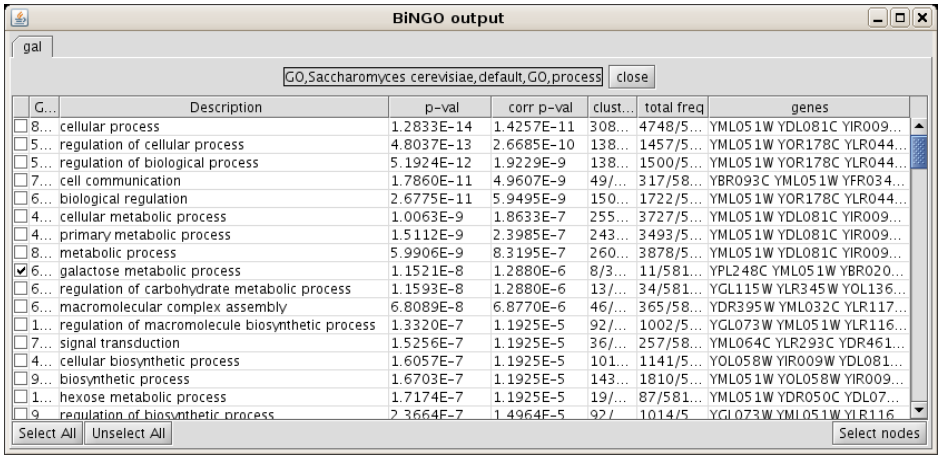


Fig. 7.17: Cuadro de resultados de BiNGO donde aparecen las categorías de GO sobre representadas.

7.7. Obtención de nuevos identificadores

Varias extensiones para Cytoscape necesitan identificar sin ambigüedades los nodos que representen genes o proteínas. Esto significa que dichas extensiones le pedirán que indique algún identificador particular. Algunos ejemplos de identificadores comunes son Entrez gene id, UniProtKB o Ensembl. Los símbolos y nombres de los genes y las proteínas usualmente no son válidos para las extensiones por la cantidad de sinónimos existentes. Por esta razón es importante saber como obtener, automáticamente, diferentes identificadores. Para ello vamos a usar una de las técnicas disponibles que emplea BioMart (Haider *et al.*, 2009). BioMart es un manejador de datos que ha sido adoptado en lugares como Ensembl, UniProt, Gramene entre otros. Para mayor información ver http://www.biomart.org

La red que estamos usando contiene un tipo de identificador proporcionado por SGD (Saccharomyces Genome Database http://www.yeastgenome.org). Sin embargo, este mismo identificador es también usado por Ensembl. El objetivo es obtener el mapeo de esos identificadores a los identificadores de UniProtKB.

- Vaya a **File → Import → Import attributes from BioMart**

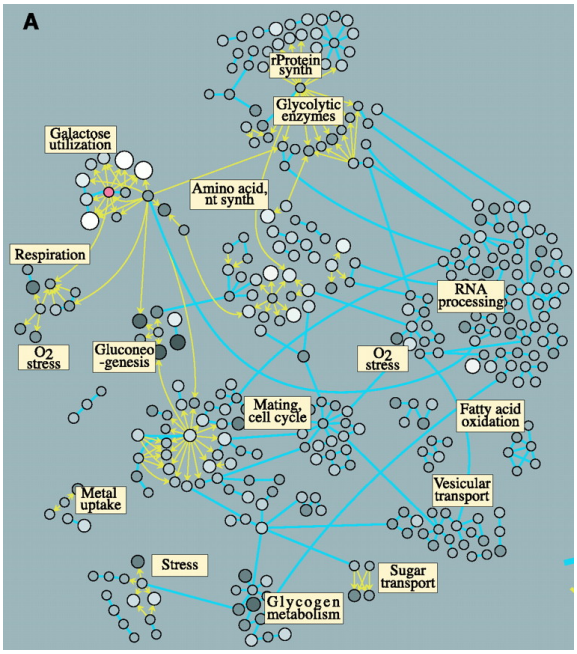


Fig. 7.18: Red del uso de galactosa. Imagen original del articulo de IDEKER *et al.* (2001) sobre la misma red usada como ejemplo en el tutorial. Si bien, algunas extensiones de Cytoscape permiten poner etiquetas a diferentes secciones de una red, lo mejor para obtener una imagen como esta es creando la visualización de la red con Cytoscape y luego agregarle las etiquetas con algún otro programa de edición de imágenes.

- Como los datos que tenemos son de levadura, buscamos en BioMart la base de datos apropiada, en este caso **ENSEMBL 55 GENES (SANGER UK) Saccharomyces cerevisiae genes (SGD...**. Ver Figura 7.19.
- En la sección **Key Attribute** seleccionamos **ID**, que es el atributo existente en nuestra red e indicamos que se trata de un identificador proporcionado de **Ensembl Gene ID**.
- Los atributos que queremos obtener son **UniProt/SwissProt accession** y **UniProt/SwissProt ID** que se encuentran casi al final del listado de posibles atributos.
- Haga clic en **Import.**
- Los nuevos atributos se pueden visualizar en el panel de datos.

7.8. Análisis de interacciones de dominio usando DomainGraph

Para la mayoría de las interacciones proteína–proteína se desconocen los detalles que gobiernan dicha interacción, por ejemplo no se sabe con que fuerza se unen las proteínas o que parte de su estructura está involucrada en la interacción. Usando el plugin para Cytoscape llamado

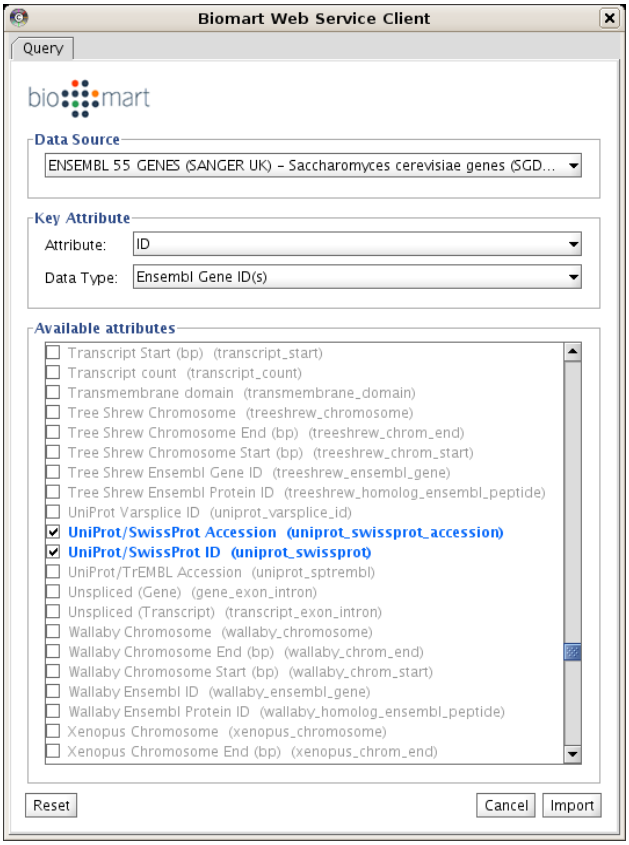


Fig. 7.19: Ventana de importación de atributos de BioMart. BioMart permite agregar una gran cantidad de información a la red. En este caso estamos usando BioMart para obtener el mapeo de identificadores de Ensembl a identificadores de UniProt.

DomainGraph (Emig *et al.*, 2008) es posible convertir una red de interacción de proteínas en una red de potenciales interacciones de dominios. DomainGraph cuenta con una extensa base de datos de interacciones dominio–dominio, tanto predichas como experimentalmente verificadas, obtenidas de la literatura, que son usadas para crear el nuevo grafo. En el siguiente tutorial solo introduciremos la funcionalidad básica de DomainGraph puesto que esta extensión también permite estudiar la interacción de variantes (splice variants) de las proteínas usando datos de expresión de exones.

Debido a que DomainGraph requiere de una base de datos local, lo mejor es instalar solo los datos para la especie con la cual se está trabajando, de otra manera se descargarían muchos datos innecesariamente.

- Vaya a **Plugins → DomainGraph → Manage Domain Graph Database → Import data for selected species into database**

- Seleccione ***Saccharomyces cerevisiae***.

- Cytoscape comenzará a descargar los datos necesarios. Este proceso tardará unos minutos. Mientras espera guarde la sesión en la cual está trabajando yendo a **File** → **Save**.

Una vez cargados los datos hay que preparar la red. Puesto que DomainGraph expandirá el número de interacciones al incluir las interacciones entre dominios lo mejor es usar una red pequeña.

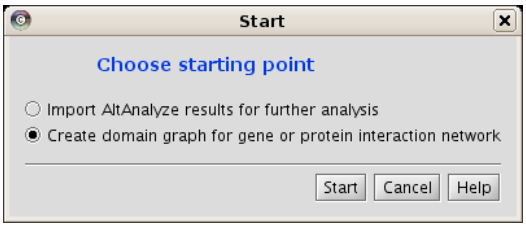


Fig. 7.20: Cuadro de diálogo de inicio de DomainGraph

- Cree una nueva red que solo contenga interacciones proteína–proteína siguiendo unos pasos similares a los de la sección 7.4.1.
- Seleccione una pequeña red que solo contenga unos 5–10 nodos. Por ejemplo aquella en que *gal4* y *gal80* hacen parte.[6]
- Cree una nueva red usando esta selección yendo a **File** → **New** → **Network** → **From selected nodes, selected edges**
- Vaya ahora a **Plugins** → **DomainGraph** → **Start DomainGraph** para comenzar a usar la extensión.
- En la ventana que aparece seleccione **Create domain graph for gene or protein interaction network**. Continúe haciendo clic en **Start** (Figura 7.20.
- Aparecerá una nueva ventana como la que se muestra en la Figura 7.21.
- Seleccione la nueva red pequeña que acaba de crear en el primer menú desplegable.
- DomainGraph contiene una larga lista que contiene diferentes datos sobre interacción de dominios. IPFAM y 3DID están basados en estructuras 3D de complejos proteicos y contienen por lo tanto los datos más confiables pero a la vez menos extensos. Los otros juegos de datos contienen principalmente predicciones de dominios [7] Para este ejemplo vamos a utilizar **INTERDOM** que es una de las fuentes de interacción de dominios más extensa.
- Haga clic en **Submit** para continuar.

[6] Si selecciona una red muy grande, DomainGraph tardará un tiempo sustancial en generar la red de interacción de dominios. Además, si la memoria en su equipo no es suficiente, puede ser que DomainGraph no llegue a completar su trabajo.

[7] Para mayor información visite http://domaingraph.bioinf.mpi-inf.mpg.de/version2.01.php.

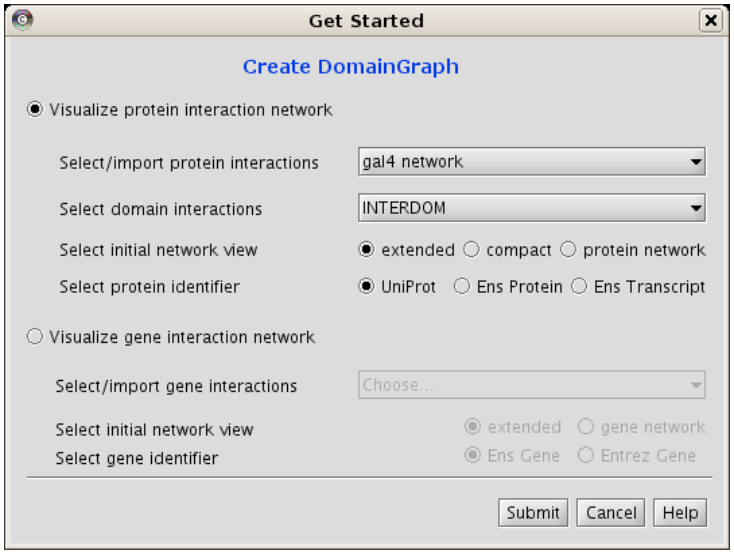


Fig. 7.21: Ventana de selección de opciones de DomainGraph.

- Ahora debe seleccionar el identificador apropiado para que DomainGraph reconozca los genes que hacen parte de la red. Después del mapeo realizado en la sección anterior ya tenemos los identificadores de UniProt requeridos por la extensión. En la nueva ventana seleccione el atributo **UniProt/SwissProt Accesssion-TOP** como se muestra en la Figura 7.22.

- DomainGraph creará una nueva red donde podrá observar las interacciones dominio–dominio de la red que seleccionó (Figura 7.23)

7.9. Enlaces de interés

Aparte de la funcionalidad y las extensiones presentadas en el tutorial, existen muchas más que no se abordaron. Además, constantemente aparecen nuevas extensiones que continúan mejorando la funcionalidad de Cytoscape. En la página de Cytoscape (ver http://cytoscape.org/) encontrará acceso a los últimos anuncios y podrá ver el listado completo de extensiones vigentes.

También, otros tutoriales en linea de Cytoscape pueden encontrarse en

http://cytoscape.org/tut/tutorial.php.

Generalmente cada una de las extensiones de Cytoscape está acompañada de un artículo científico y de un portal que incluye un tutorial para el uso de la extensión. En la página de *plugins* del portal de Cytoscape aparecen las citas a los artículos y el enlace a los portales de las extensiones.

El artículo de Cline *et al.* (2007) contiene valiosa información sobre varios de las extensiones que no fueron mencionadas en este tutorial.

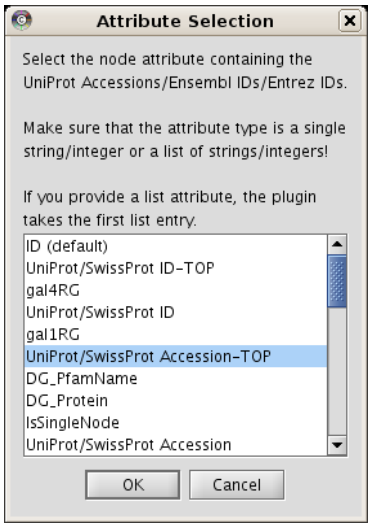


Fig. 7.22: Ventana de selección del atributo que contiene el mapeo a identificadores UniProt de la red

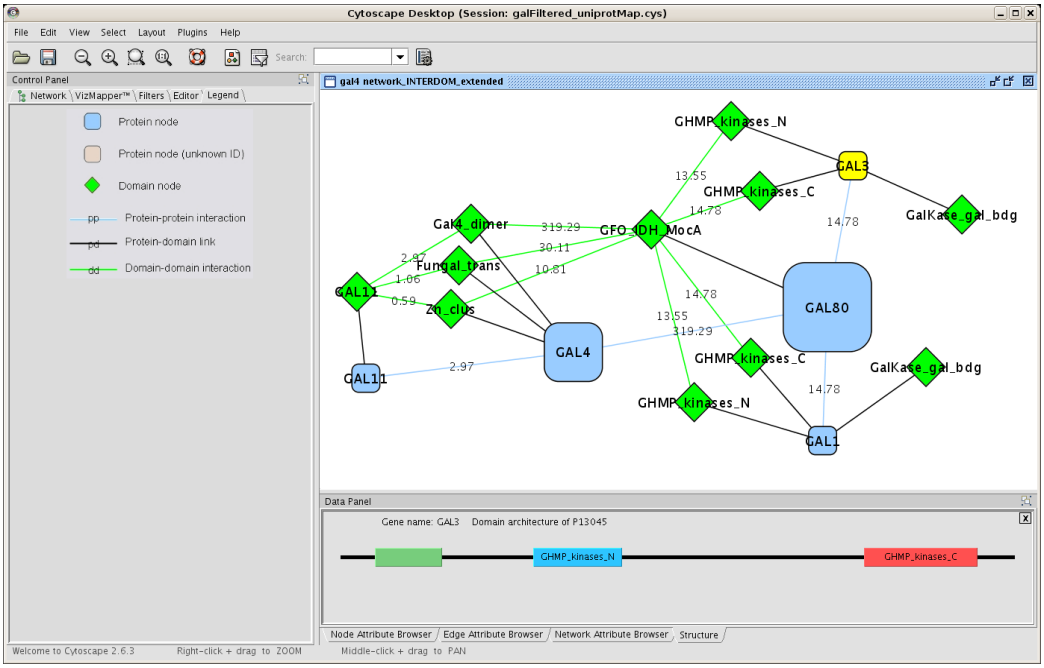


Fig. 7.23: Descomposición de una red de interacción de proteínas en una red de interacción de dominios usando DomainGraph. Todas las posibles interacciones de dominios que explican una interacción de proteínas aparecen en la nueva red que crea DomainGraph. Al llevar el ratón sobre los nodos que representan las proteínas, aparece un letrero que lista los dominios de esta. Al hacer doble-clic sobre el nodo se muestra gráficamente la arquitectura de los dominios de la proteína seleccionada en el panel de datos. Igualmente, al llevar el ratón sobre los dominios aparece una lista con otras proteínas que contienen dicho dominio. Los números sobre las aristas indican la confidencia acerca de la interacción entre dos dominios. En esta red se puede observar que *GAL4* contiene tres dominios, Gal4_dimer, Fungal_trans y Zn_clus. *GAL80* contiene un solo dominio llamado GFO_IDH_MocA. La interacción *GAL4–GAL80* puede ser explicada por cualquiera de los dominios de GAL4 con el dominio GFO_IDH_MocA. Sin embargo, la interacción de los dominios GAL4_dimer y GFO_IDH_MocA es la más robusta con un puntaje de 319.29.

8

Bioinformatics for proteomics tutorial of practice session

Flavia Vischi Winck

8.1. General introduction

8.1.1. Bioinformatics in protein and proteome studies

Proteins are macromolecules involved in several different biological processes. In order to study the identity and to characterize these proteins an understanding of their role in biological systems is required.

Many computational programs were specially developed to help characterize and identify proteins. Those tools are essential for the large-scale analysis of proteins and have been continuously updated in order to refine and to improve the *in silico* characterization and identification of proteins.

General instructions

Our data analysis will focus on: computing physicochemical properties of proteins based on protein primary structure; protein molecular homology modeling (or comparative modeling); and identification of proteins by Peptide Mass Fingerprint (PMF) and tandem Mass Spectrometry (MS/MS).

The practical sessions will give you an overview of the bioinformatic tools used for proteomic studies. Those sessions are the first step into the bioinformatics for proteomics field. It is strongly recommended that you explore the programs presented here in more detail in order to decide which applications are best suited for your research area.

All the protein sequences and other additional information related to this course will be located in your computer in the folder named "Bioinformatics for Proteomics." Please download the content of this folder to your local directory.

8.2. Prediction of physicochemical properties of proteins

8.2.1. Introduction

The prediction of the physicochemical properties of proteins is useful where a protein sequence is not available in a database. The determination of these properties can guide experimental processes, such as chromatographic analysis, purification, determination of protein purity, and protein extraction. These analyses only require the primary sequence of the protein, which can also be deduced from the gene sequence or transcript sequence.

8.2.2. *Computing physicochemical properties of proteins*

The physicochemical properties of the proteins are determined by a group of characteristics of the molecule, including total charge state, amino acid composition, presence of additional functional groups, hydrophobicity, isoelectric point, molecular mass, and others. The genomic sequence of many different species is now available and the use of computational programs for data analysis is essential.

For protein analysis, information in protein databases can be used to predict certain properties about a protein, which can be useful for its empirical investigation.

Protein analysis tools can predict many properties of the proteins, including molecular weight (mW), theoretical isoelectric point (pI), amino acid composition, atomic composition, extinction coefficient, estimated halflife, instability index, aliphatic index and grand average of hydropathicity (GRAVY). Other tools can also predict enzymatic cleavage sites, presence of post-translational modifications and isotopic masses of peptides. There are many programs available for those applications; some examples are presented in Table 8.1

Table 8.1: Predictors. Examples of bioinformatic tools for prediction of the physicochemical properties of proteins.

Tool/software packages	Website
The sequence analyzer	`http://www.proteinchemist.com/progs.html`
ProtParam	`http://www.expasy.ch/tools/protparam.html`
Protein Calculator	`http://www.scripps.edu/cdputnam/software/software.html`
Emboss-Lite	`http://helixweb.nih.gov/emboss_lite/protein.html`

One of the most commonly used predictor tools is located in the **Ex**pert **P**rotein **A**nalysis **S**ystem (Expasy), at `www.expasy.ch`. This server contains many other tools for protein sequence analysis based on the protein primary structure.

Let's have a look on this website.

1. Open the website `www.expasy.ch` which is shown in Figure 8.1.

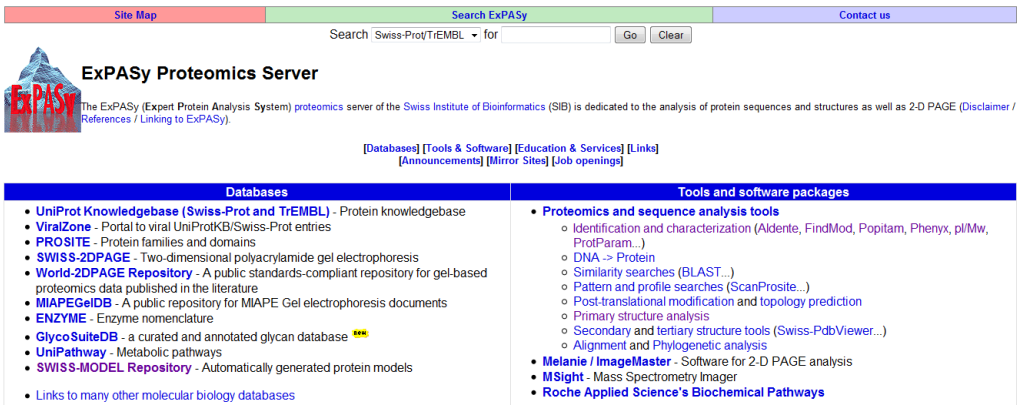


Fig. 8.1: Expasy website. Several tools for proteomic studies are located in this server.

This website contains many of the tools needed for extracting and interpreting the biological meaning of a protein primary sequence. Here you also can find many other tools and general information (e.g. courses, databases, jobs and external links).

2. In the session "Tools and software packages", open the session Proteomics and sequence analysis tools

3. Here you can find the many others applications related to protein sequence analysis, including: *[Protein identification and characterization] [DNA → Protein] [Similarity searches] [Pattern and profile searches] [Post-translational modification prediction] [Topology prediction] [Primary structure analysis][Secondary structure prediction] [Tertiary structure] [Sequence alignment] [Gateways] [Phylogenetic analysis] [Biological text analysis]*

4. Click on **"Primary structure analysis"** on the top of the page, or scroll down the page and find the same session.

5. **Click on "ProtParam".** This tool will enable you to calculate the many different physicochemical parameters of a specific protein. Here you will see the page for inserting your protein sequence. You can either enter the accession number of your protein if it is located in the "SwissProt" database, or you can paste the amino acid sequences in the area indicated in the Figure 2.

6. Open the folder named "Protein sequences" (../Bioinformatics for Proteomics/Protein sequences). Inside this folder you will find a file named "ATE2F2_ATH.fasta". Open this file and copy the amino acid sequences to clipboard. Be sure to only copy the amino acid sequences without the protein description text.

7. Paste the sequence in the paste area of ProtParam as indicated below in the Figure 8.2.

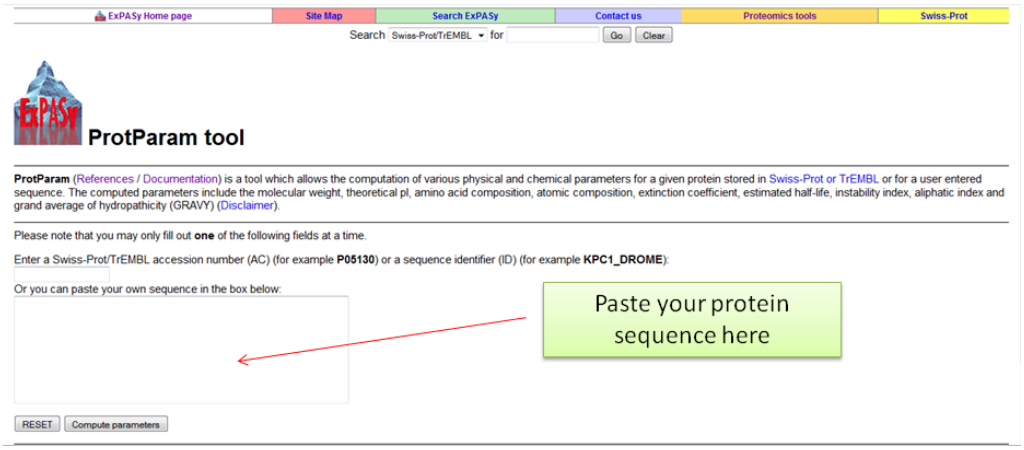


Fig. 8.2: Protparam proteomic toll. Paste area of the software ProtParam in the Expasy server.

8. Press "Compute parameters". Check the results. For detailed information on how each parameter was calculated refer to the file about **References/Documentation** in the website or in your local folder (../Bioinformatics_for_proteomics/Protein characterization/expasy_tools05).

9. Save the results page by saving the file in text format or html format. You can also obtain the amino acid composition on a CSV (comma separated values) format by clicking on the link in the result page and saving it.

This is one of the many ways you can perform the calculation of the physicochemical parameters of one single protein sequence. There are other programs available for calculating the same or extra parameters; some examples are shown in the Table 8.2

8.3. Protein molecular homology Modeling (or comparative protein modeling)

(This part of the tutorial is based on material created by Janusz Bujnicki, who presented the original tutorial in the ECCB09. For re-distribution of this material, please send an inquire e-mail to vischiwi@uni-potsdam.de).

8.3.1. Introduction

While much useful information about a protein can be derived from looking at both DNA and protein sequence information, the only way to fully understand all of its function is by looking

Table 8.2: Predictors and their computational tools. Different tools can predict different protein properties, and are usually complementary.

Physicochemical property predicted / Tools and predictions available				
	ProtParam	Protein Calculator	The sequence analyzer	Emboss-Lite
Isoelectric point	X	X	X	X
Molecular weight	X	X	X	
Hydrophobicity	X		X	X
Half-life	X			
Extinction coefficient	X		X	
Aliphatic index	X			
Aminoacid composition	X	X	X	
Atomic composition		X		
Instability index	X			
Titration curve			X	
UV spectrum		X	X	
Solvent content		X		
Proteolytic products		X		X
Total free energy				X

at its 3D structure. Proteins operate in a 3D environment, and by looking at them in this context it is possible to explain occurrences that could not be accounted for by other analyses. The information contained in the primary protein structure (amino acid sequence) governs how the secondary and tertiary protein structures will be folded. How the atoms interact to form a certain protein structure and how those atoms interact with other surrounding molecules will define the biological function of a protein.

In 1961, Anfinsen and colleagues performed the first clear demonstration on how proteins are folded.

"... information... for the assumption of the native secondary and tertiary structures, is contained in the amino acid sequence itself". Anfinsen *et al.* (1961).

Anfinsen realized that the proteins are folded in a "step-by-step" way. Many different intermediate states can play important roles in determining the final fold composition of a protein, as mentioned in the text bellow:

"The results rule out the sequential formation of one active molecule after another. They suggest as a major possibility that some disulfide bonds formed during the early stages of oxidation are not identical with those of the native protein but undergo rearrangement to yield the native configuration" (Anfinsen *et al.*, 1961).

For protein molecular homology modeling some tools are needed for visualization, comparison, superposition, alignment and modification of protein structures. In our course we will use the programs Deep View, formerly called SwissPDB viewer (`http://spdbv.vital-it.ch/`), BioEdit (`www.mbio.ncsu.edu/BioEdit/bioedit.html`) and Chimera (`http://www.cgl.ucsf.edu/chimera/`).

First Instructions

-Gmail-

e.mail: `curso.bioinfo.2009@gmail.com`

username: mexico09

-Genesilico Metaserver and Frankenstein tool-
Username: curso.bioinfo.2009
Password: mexico09

-swiss model-
Username: curso.bioinfo.2009@gmail.com
Password: mexico09

Today we will see how to perform a basic project for protein comparative modeling. We will partially model a plant transcription factor of *Arabidospsis thaliana.* The model produced here can be further improved and refined. Our target protein sequence is a plant transcription factor. There is no structure experimentally determined for this protein. For modeling this protein we will make use of the MetaServer (`www.genesilico.org`), which contains many other servers for structure and sequence analysis.

The amino acids sequence of our target protein is located in the folder "Protein sequences" and it is named "ATE2F2_ATH.fasta" or "AT2F2_ATH.txt".

Keep the files you generate during the practice in a local home/user folder. The files are also located in a folder named "Modeling" in case you need them to complete some tasks.

Predicting the fold of the target protein

1. Go to the URL `https://genesilico.pl/meta2/`.
2. Press OK if prompted to accept the certificate.
3. Log in to the course account. Login: curso.bioinformatica.2009, Password: mexico09

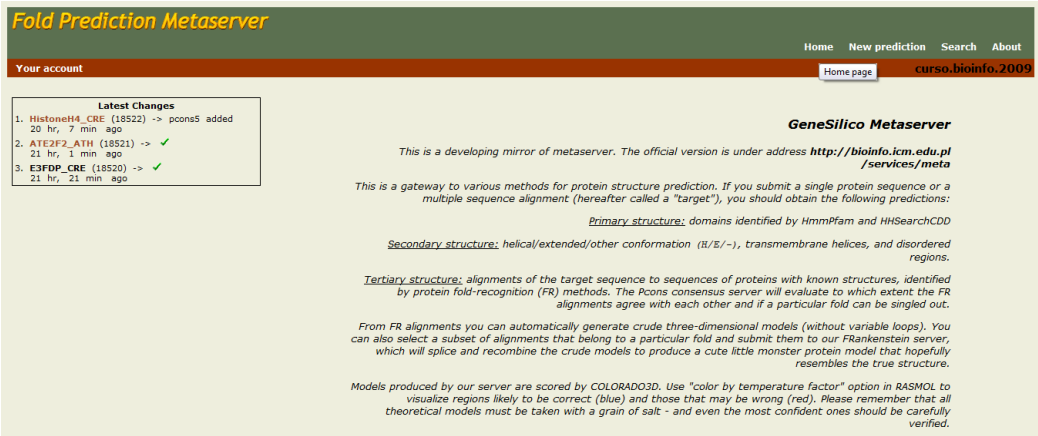


Fig. 8.3: Genesilico Metaserver. A multi server for protein folding recognition.

4. Click on the "New prediction" link. The page for submitting sequences for fold recognition (FR) analysis should be opened.

5. The FR analysis for ATE2F2_ATH is a lengthy process, so it has already been completed. For this reason we will not run the ATE2F2_ATH job again, but rather use the saved results for ATE2F2_ATH.

6. Click on the Search link and include the name ATE2F2_ATH in the job name area.

7. Press Search button.

8. Click on the link ATE2F2_ATH, which should appear on the left corner of the page (be careful to do not delete the entry).

9. After clicking the ATE2F2_ATH link, the page containing all FR results should open. (You can also find the local copy of this page in ../Bioinformatics for proteomics/Modeling/Fold _Recognition_ATE2F2_ATH.htm).

10. Analyze the results of each server (e.g. hmmpfam, blastp, etc.) by selecting the template sequence. Try to answer those questions:

    - What is the predicted secondary structure pattern?
    - Did any of the primary fold recognition servers report an alignment with a significant score?
    - Do the fold recognition methods agree with each other? Is there a "consensus" fold according to the rankings made by Pcons5

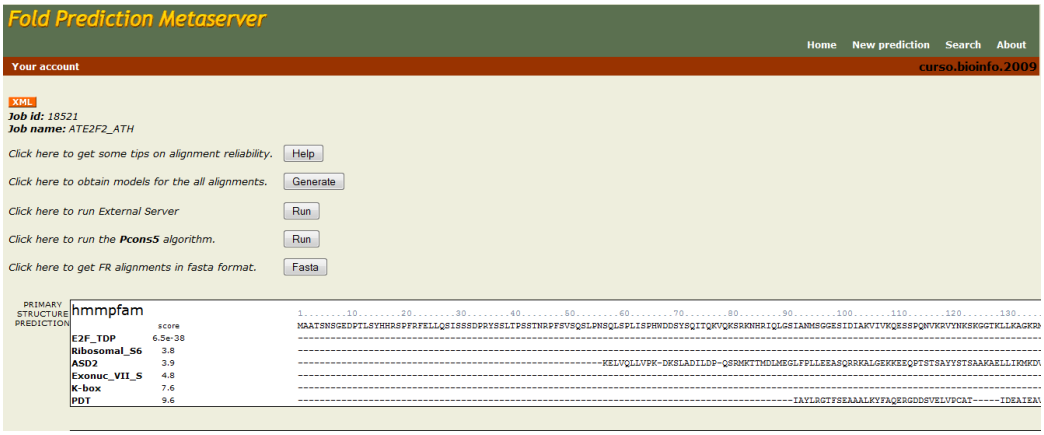


Fig. 8.4: Genesilico Metaserver Fold-recognition. Results page of fold recognition (FR) analysis done simultaneously in different servers.

Selecting the template

It is usually better to use multiple alignment analysis, rather than a single alignment analysis, to identify the best protein template. Select those with the highest identity in pcons5, the highest score in blastp and the most complete alignment.

Now we will download the top templates from the Protein Databank (PDB) website.

1. In the FR results page, look for the "blastp" alignment box and right-click on the PDB code (1cf7 A) in the MetaServer and open a link to the PDB database in a new tab or window.

2. The PDB entry for the given template structure will open (Figure 2). Read the summary information. Note what the name of the protein is and if it contains any ligands (e.g. ions, DNA).

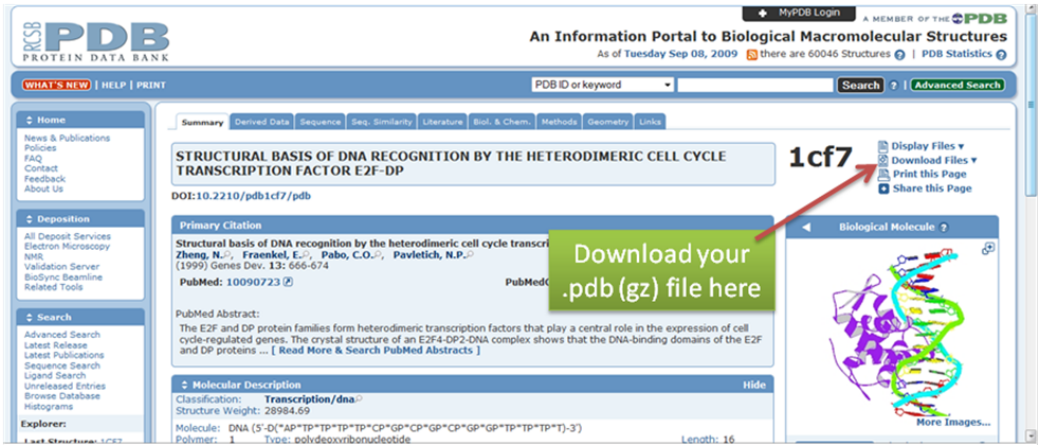


Fig. 8.5: Protein Data Bank (PDB). PDB contains the information of the tridimensional structure of thousands of proteins.

The PDB contains detailed information about the protein structures of thousands of proteins. It is important to have information about the main details of the protein structure. Data about the method of determination of the structure (X-Ray or NMR), resolution (Angstrons), solvents, presence of binding elements and secondary structure are available for each protein contained in the databank.

Try to explore the submenus in the upper part of the website and navigate through them. Good information can be found in Derived data, Sequence, Biol & Chem., Methods, Geometry and Links.

When multiple structures are available for a given protein you should select the one that is solved in a similar environment. If a model is a ligand-bound form, the template solved with ligand is preferred over the ligand-free structure. Also, X-ray structures are preferred over NMR structures. Use structures that are solved with higher resolution. It is best to use protein chains without missing electron density.

3. Click on "Download Files" in the icon right next to the structure code to download the PDB file of the structure 1cf7 to your local folder. This file is also contained in the local folder (..Bioinformatics_for_proteomics/Modeling/1cf7.pdb.gz).

After getting the template structure of the protein in the PDB website, we will take a more indepth look into the template structure with the program DeepView.

Looking at the template in DeepView

DeepView, also called Swiss-PDB-Viewer (`http://spdbv.vital-it.ch/`), is a versatile program for visualization and manipulation of macromolecular structures (Guex and Peitsch, 1997). It can be used for creating a "project" file to hold the template structure and target sequence, and that allows for convenient creation and editing of the target-template alignment. The project can be sent to modeling servers that can automatically parse out the template structure and target-template alignment and use them directly for building a 3D model of the target. You can learn more about the program here: `http://spdbv.vital-it.ch/main_tut.html`.

DeepView is a free package written by a team from the Swiss Institute for Bioinformatics. It provides a variety of visualization options as well as allowing homology modeling and energy minimization. It also interacts with the POV-Ray raytracing package to produce publication-quality molecular graphics.

Now we will see how to use the visualization components of DeepView to analyze both single and multiple structures.

This program is already installed on your computer for our practical session and can be downloaded from `http://spdbv.vital-it.ch/`

Basic DeepView Controls

Starting DeepView

DeepView is started by double-clicking the icon after installation.

You should see a splash screen appear.
Clicking on this will make it go away, and will leave you with the main program toolbar

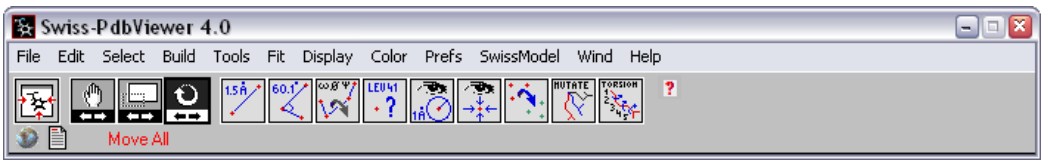


Fig. 8.6: Main toolbar. Contains basic controls for editing the structures.

Opening Files

If the PDB file was generated by DeepView, then your original colors and view will be preserved. If not, a default view will be presented.

1 - Select File → Open from the File menu, and select the file. Open the file named "1cf7.pdb" located in your local folder or in the folder "Modeling" (../Bioinformatics for proteomics/Modeling).

Press OK if you are prompted about "unknown groups" and missing "CONNECT" information. The molecule will appear in the display window. Don't worry if you see different colors or different chain representation than that on the figure below—the current look depends on your Swiss PDB Viewer settings. A pop-up window will appear and must be closed.

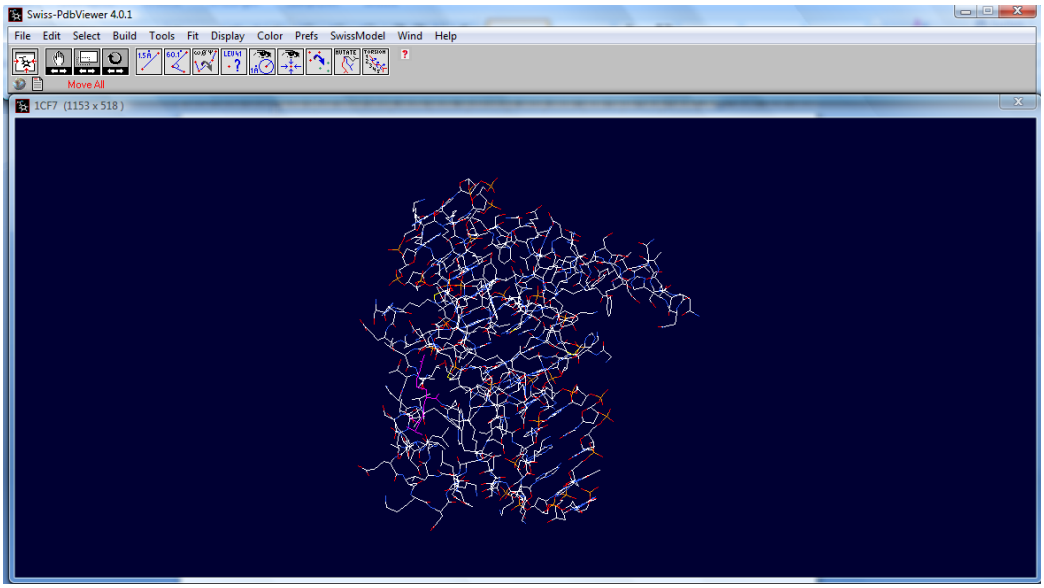


Fig. 8.7: Deep View. Structure of a protein in 3D.

Basic Viewing Controls

When you first open a molecule in DeepView you should have the main toolbar open, and attached to this should be a viewing window containing your molecule. You may have other windows open as well, but we'll ignore those for the moment and come back to them later. The first thing you'll want to do is to be able to move the molecule around.

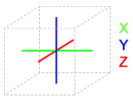
Rotate, Translate and Zoom

The 3 basic controls for moving a 3D molecule are rotate, translate and zoom. These can all be accessed by using a mouse. To use these controls, position your mouse within the molecule window and use the controls shown below. For each control there is also a button placed on the main toolbar that can be used to achieve the same manipulation.

Movement	Mouse Control	Description	Toolbar Button
Rotate	Left mouse button	Up/Down rotates about the x-axis. Left/Right rotate about the y-axis.	
Translate	Right mouse button	Simply drag the molecule in the direction you wish to move it.	
Zoom	Both mouse buttons	Move up to zoom out, move down to zoom in.	

1- Play with the controls and make the molecule big enough to see individual atomic interactions.

Although these controls give you a fairly flexible system of movement, they do not provide the fullest range possible, as they do not allow you to perform rotations or translations about the Z-axis (the one coming directly at you out of the screen). To achieve this extra movement, there are a series of keyboard modifiers you can use in combination with the controls shown above.

F5 = Restrict movements to the X-axis **F6** = Restrict movements to the Y-axis **F7** = Restrict movements to the Z-axis	

Center View

A very useful tool is the far left button on the toolbar. This button will adjust the translation and zoom of your molecule so that it is completely visible, and sits in the middle of the screen. It will also adjust the center of rotation so that it falls within the middle of whatever is visible.

1 - Press the center view button located in the main program toolbar.

Slab

One problem with looking at 3D structures is that it's difficult to see what's going on in the middle of the structure, because exterior residues obscure interior residues. To work around this issue, you can use the Slab display mode. This shows you an interactive slice through your protein, and slices are only displaced where a residue falls inside the slice.

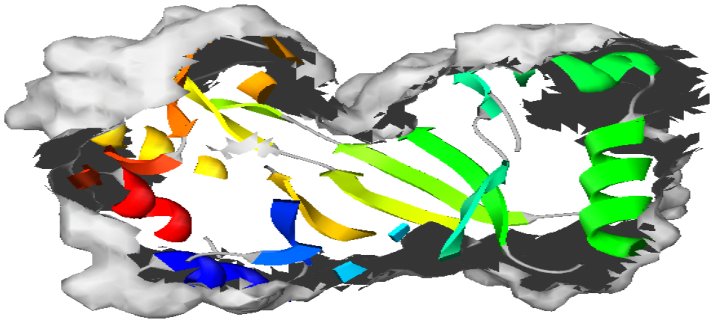


Fig. 8.8: Slab image. Here you can see different slices of the protein structure.

1- Turn the Slab display on and off by selecting Display → Slab from the main menu toolbar. Once you are in the Slab view all the movement controls work the same way as before. One extra refinement is that if you want to drag your molecule through the slab (so you can see residues at the front or back) then you should use the Translate tool with F7 held down (to restrict movement to the z-axis).

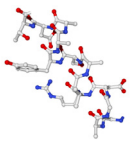
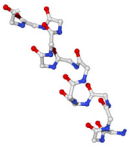
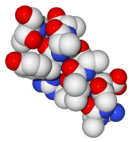
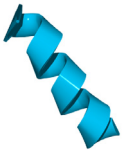
Different Structural Representations

Some of the most common representations are shown below.

The Control Panel

1- Open the menu Window → Control Panel from the main toolbar.

The control panel shows the aminoacid residues and some basic information about the struc-

	This is a standard “ball and stick” representation of a stretch of protein. All the atoms in the structure are visible as points in space or small spheres, and the covalent bonds joining them are shown as lines or tubes.
	This is a similar representation to the ”ball and stick,.[ex]cept that it shows only the backbone residues. The atoms in the sidechains of the various amino acids are omitted. This produces a cleaner, less cluttered model in which larger structuresare easier to visualize.
	This is a space-filling representation. Covalent bonds are not shown, and atoms are represented by larger spheres, which are sized to represent the scale of the electron shell around the atom. This is a more realistic view, but is only useful when applied sparingly, as it quickly obscures the view of other parts of the model.
	The final representation is the ribbon. This is a very simplified view where neither atoms nor bonds are shown. Instead a tube structure runs along the path traced out by the carbon backbone of the protein. Where the tube passes through a recognised region of secondary structure it is flattened to a ribbon shape. This representation is very useful for looking at larger structures within a protein

ture you are looking at. It contains the name of the current structure and whether it is visible and able to be moved.

A description of the columns present in the control panel is listed in the table below.

You will see that next to some residues there is a tick mark in some of the columns. Where this mark is present means that the corresponding attribute is “turned on” for that residue. All of these attributes can be toggled by clicking on them with the left mouse button.

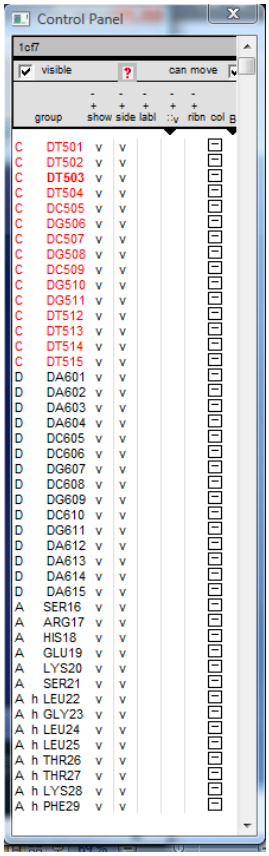


Fig. 8.9: Control Panel

In the control panel, you can see a brief and abbreviated description of each residue. The formatted string for the group identifier is laid out as show below.

On two of the columns (Surface and Color) there is a little triangle ∇ (below the column name). This indicates that the column can show more than one set of information. If you click on the triangle, a list will drop down and display the different options for that column.

Working with Selections

Often when looking at a molecule you need to work with a subset of the residues within it. In order to do this you need to be able to select some residues, either individually or on the basis of some property calculation.

Changing Colors

DeepView allows you to alter the color of your molecules. There are a number of built-in color

Columns Headers	Description
Group	This text provides a description of a residue in your structure. It contains information about chain, secondary structure, amino acid and position.
Show	This column denotes whether the backbone atoms of each reside are visible or not.
side	This column indicates whether the residues side chain is visible. By default a side chain is never visible unless the backbone is also visible.
Labl	Turning this option on will place a small floating label next to the residue. By default this label will show the amino acid name and position.
: :	This interestingly named column actually shows whether or not each residue in the structure has a surface associated with it or not.
Ribn	This says whether a ribbon representation of the structure is present for each residue
Col	This last column indicates what color is being applied to each reside. Usually this is filled with a block of color, but can also be a white box with a "-" sign in it, indicating that the object's default color is being used.

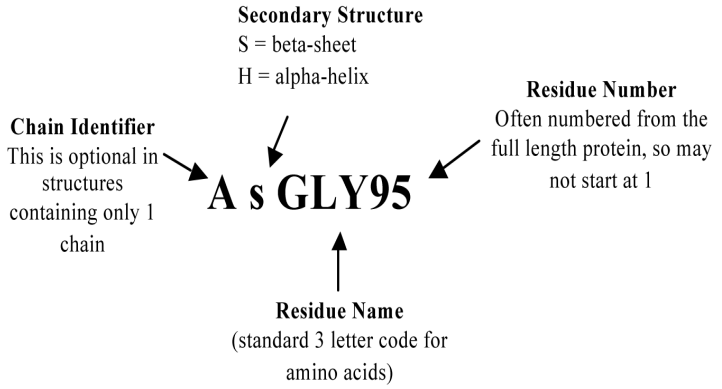


Fig. 8.10: Residue description. The residue descriptor shows some characteristics of the residues in the control panel.

schemes, or you can define your own. Also, you can apply separate color schemes to each different structural representation of your molecule.

Standard Color Schemes

The standard color schemes in DeepView are found under the "Color" menu on the main toolbar, in the first division at the top. They mostly color your molecule according to the physicochemical property of its residues. A full description of the most useful color schemes is shown below.

Making Selections with the Control Panel

Scheme Name	Colors
CPK	Colored per atom by element type: Carbon White Oxygen Red Nitrogen Dark Blue Sulphur Yellow Hydrogen Light Blue
Type	Colored according to residue properties: Acidic Red Basic Blue Polar Yellow Hydrophobic Grey
Secondary Structure	Colored according to secondary structure Helices Red Sheets Yellow Loops Grey
Secondary Structure Succession	Loops are colored grey. All main secondary structure elements are colored from cold (blue) to hot (red) running from the N to the C terminus.
Chain	Colors each chain in a structure a different color. Useful if you are looking at a complex of two or more molecules.
Accessibility	Colors the structure according to the solvent accessibility of the residues. Cold colors indicate that residues are inaccessible, hot colors mean they are solvent exposed.
Other Color	Brings up a color selector and allows you to pick any color you want.

In addition to the selection tools in the Select menu, you can also make selections using the group column in the Control Panel. In the Control Panel, all selected residues, however they were selected, have their names highlighted in red. Clicking on a chain identifier will select all the residues in that chain.

Applying Attributes to Selections

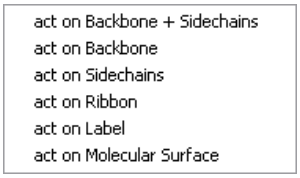
Once you have made your selection you will be able to work with those residues. If you already have some attributes turned on, and you just want to add or remove some new residues, you can do this by clicking on the + and ˇ symbols above the column name. This will leave your existing attributes unaffected but will turn the selected attribute on or off.

Selecting areas for changing colors

To apply a color, select the appropriate entry from the Color menu. A common problem occurs when a ribbon representation showing and a user selects a new color scheme, and nothing happens. This likely happens because the user has only changed the color of the backbone and side chains, which is not visibly apparent.

DeepView allows you to apply a separate color scheme to each of the following:

Go menu Color→Act on XXXXXX (where XXXXXXis what you are currently applying colors to) and select the part of the structure to which you want to apply the color.

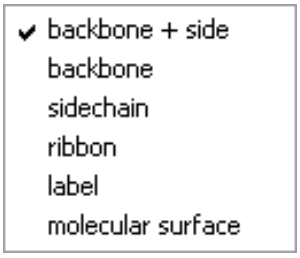


Changing Colors in the Control Panel

In addition to the method outlined above, you can also use the Control Panel to change the color scheme for your molecule.

To change where colors are applied, click on the little ∇ symbol just to the right of the Col column. This will bring up a little pop-up menu from which you can choose what you're working on. The letters over this symbol indicate what you're working on at the moment (BS = backbone and side chain, R=ribbon etc.).

You can change colors by using the colored blocks in the Col column. These work in the same way as all the other Control Panel columns in that if you click in a box with the left mouse button, you change the color for one residue. If you click in a box with the right mouse button, you change the color for all residues.



Levels of OpenGL

There are three different levels of graphical complexity within DeeView and increasingly amounts of your molecule will become solid as you go through them. The details of what is solid in each level are shown below.

	Atoms	**Bonds**	**Surfaces**	**Ribbons**
Wireframe	no	no	no	no
3D	no	no	yes	yes
Solid 3D	yes	yes	yes	yes

1- Change the display settings under the "Display" menu. Select "Render in Solid3D". This will also turn on "Render in 3D", which will stay on even when solid 3D is turned off again.

Using the Select Menu

Under the "Select" menu there are tools that allow you to make selections on the basis of various functional criteria. The most commonly used options within this menu are detailed below.

Name	**Selection Criteria**
All	Selects all residues in the current molecule
None	Deselects all residues in the current molecule
Inverse	Inverts the current selection
Visible Groups	Selects everything that is currently visible (either as a backbone, sidechain or ribbon). This selects everything that is theoretically visible, even if it isn't actually currently visible in the molecule window.
Group Kind	Selects all of a particular amino acid, DNA base etc.
Group Property	Selects on the basis of the biophysical properties of your residues (acidic, basic, hydrophilic, hydrophobic etc.).
Secondary Structure	Selects residues based on the type of secondary structure they are part of.
Accessible amino acids	Selects residues whose solvent accessibility is greater than a user-supplied cutoff value. If you want to select residues whose accessibility is less than a certain value, then you should use this function in combination with the inverse selection function.
Neighbors of selected amino acids	Selects all the residues that fall within a sphere of user-defined radius around all the currently selected residues.
Groups close to another chain	Selects all resides that fall in the region of contact between two chains in the same molecule. The user can specify the cutoff for how close they must be.

1- On the Control panel click in the Chain identifier A and change the attributes described in the table above to see the modifications on the proteins structure.

2- Apply the label to some residues (C and D residues) by selecting them and selecting the function "labl" by clicking on the symbol + in the control panel. You will see the identifiers of each residue appear in the structure.

3- Attribute different color to each by clicking in "col" in the control panel and selecting a green color from the color pallet.

Adding Surfaces

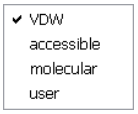
Different Kinds of Surface

DeepView offers three different kinds of surface representation. Here we will see the addition

of the Van de Walls (VDW) and molecular surfaces.

Van der Waals Surface

This kind of surface is also sometimes referred to as a space filling view. Instead of all your atoms being represented by either points in space or small spheres, in a Van der Waals representation they are expanded out to cover the full area, which would be occupied by their electron shells. Van der Waals surfaces are particularly useful because they are often used to highlight residues in an active site, or to show how a substrate would fill a binding pocket.



You can add Van der Waals surfaces to a molecule directly using the Control Panel. Just underneath the surfaces column (: :) there is a ∇ symbol. If you click on this, you get a drop down menu showing the various kinds of surfaces available. Select VDW from this list and then use the surface column in the control panel to add a surface to whichever residues you like.

In order to see a Van der Waals surface, the underlying atoms must be visible. This means that the Backbone and Side chain columns in the Control Panel must be selected for the surface to have any effect. You should note that unless you are using OpenGL + Solid 3D, the Van der Waals surface will only show up as a series of dots around your atoms.

Molecular Surface

The other main type of surface you will find useful is the molecular surface. This type of surface is usually applied to whole molecules rather than individual residues. It adds an outer skin to the molecule, stretched around an imaginary Van der Waals surface underneath. This means that only the outside of the molecule is covered, and that any other underlying molecular representations are still visible.

Molecular surfaces differ from Van der Waals surfaces in that they must be calculated before they can be displayed. Once you have calculated a molecular surface you can add and remove it using the Control Panel, the same way as for all other features. Before you calculate a surface, you should check your surface preferences in Preferences → Surfaces.

For the most part, you can leave these settings alone, but there are a couple of things to be aware of.

The main thing to check is that the surface quality is set to a sensible value. This determines how complex your surface will be. High quality surfaces take up a lot of memory and will slow things down considerably. For most purposes, you will never need a surface quality higher than

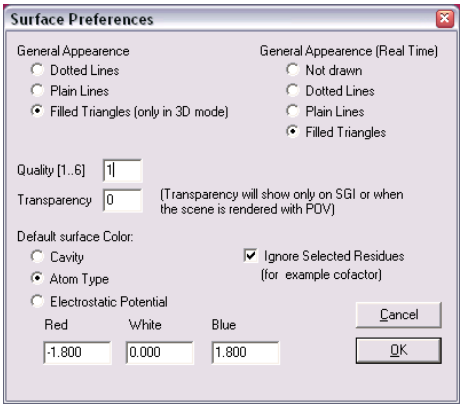


Fig. 8.11: Surface Preferences window. Selection of parameters for creating surfaces.

three, and usually a quality one or two surface will suffice.

The other option to be aware of is the one reads: "Ignore Selected Residues". This means that whenever you calculate a molecular surface, any residues that are selected when you perform the calculation will be ignored, and the surface will pass under them. Therefore, if you want to calculate a surface for a whole protein, you must use Select → None before starting the calculation. If your structure contains a substrate then you should select only the substrate before calculating the surface.

1- Calculate a molecular surface using Tools → "Compute Molecular Surface". Once the surface has been calculated you can manipulate it like any other feature using the Control Panel.

If you want to permanently delete a calculated surface, you can do this by selecting "File → Discard → Surface".

Working with Multiple Structures

By now you should have enough information to allow you to manipulate and view a single structure in pretty much any way you like. DeepView allows users to view up to 12 structures at the same time. Working with multiple related structures at the same time can provide more detailed information about the protein compared to individual viewing, as it allows you to look at the differences between related structures and relate them to known functional differences.

Loading Multiple Structures

Once you have loaded in your first structure you can use exactly the same methods to open a second one. The two structures will appear simultaneously in the molecule view window, but will be unaligned when first opened. The two structures will not be merged together, but will

be put into separate layers.

Choosing an Active Layer

A new concept you need to learn when working with multiple structures is that of the active layer. Only one layer can be active at a time. It is the active layer whose details appear in the Control Panel and on which all actions will operate (e.g. color changes, surface calculations etc.). When you have more than one layer to open, it is useful to look at the "Layers Info" window, which can be opened from the Wind menu on the main toolbar.

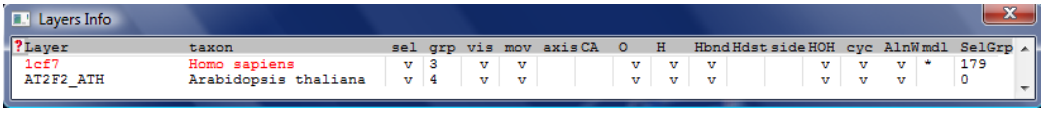


Fig. 8.12: Layers Info. Describes which layer is active or not.

This window displays a short summary of the status of the various layers currently available. The columns that you are likely to use are detailed below.

Column	Description
Layer	The layer column gives the names of the structures you have open in each layer. One of the names will be highlighted in red; this is the currently active layer. You can make any layer active by clicking on its name in this column.
Vis	This column shows whether the layer is currently visible. This option overrides any options you may have set in the columns of the Control Panel, and is the same thing as the "visible" tick box at the top of the Control Panel.
Mov	This column shows whether this layer will move when you apply any of the standard movement controls. Having this attribute set to different values for different layers allow you to alter the relative alignment between two structures.
SelGrp	This is a counter to show how many residues are currently selected in each layer.

The Active Layer can also be set using the Control Panel. At the top of the Control Panel is a small piece of text showing the layer that is currently active. If you click on this text you should see a popup menu that shows all of the available layers. Simply select the one you want to be active.

Preparing data for modeling

After loading the template structure (1cf7.pdb) into DeepView and learning a bit about the basic functions of the program, you should see the protein molecules sitting on DNA.
For the purpose of this tutorial, we will select separately protein and DNA chains and save them as two separated files.

1. Select chain C and D by ctrl-clicking on the letters C and D on the Control Panel (all residues from chain C and D should turn red).

2. Go to Menu File→ Save→ Selected Residues of Current Layer...

3. Save the file under the name 1cf7dna.pdb in your local/user directory.

4. Remove the DNA residues by going to Build→ Remove selected residues. Now you should have your structure without the DNA chains.

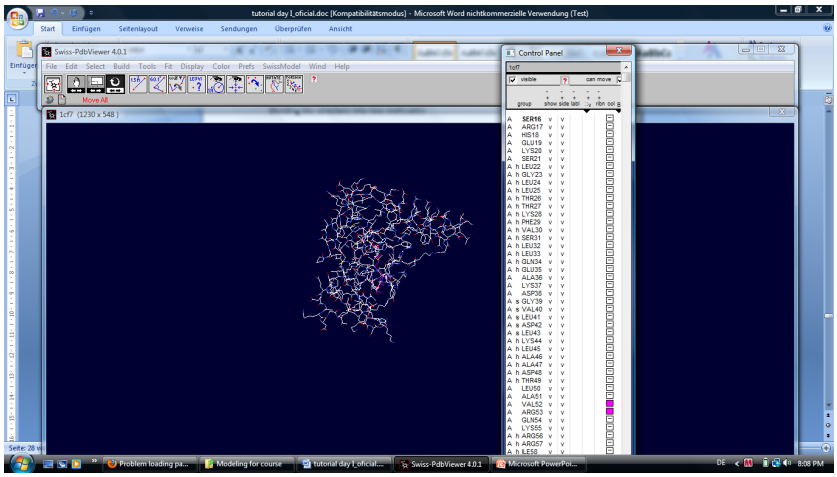


Fig. 8.13: Pre-processing images. Separation of the chains of the different molecules and deletion of DNA chains from the present structure.

5. Select chain A by left clicking on the letter A on the Control Panel and Ctrl + A (all residues from chain A should turn red).

6. Go to Menu File → Save→ Selected Residues of Current Layer...

7. Save the file under the name 1cf7a.pdb.

8. Close all windows. Go to menu File → Close.

Creating a DeepView project and building initial target template

METHOD 1. Fit raw sequence.

1. Open the structure 1cf7a.pdb from your local directory into a new DeepView window. Go to File → Open PDB file... and choose the file (1cf7a.pdb).

2. Load the target protein sequence file ATE2F2_ATH.fasta. Go to menu SwissModel→ Load Raw Sequence from Amino Acids... A window should appear for selecting the path of your file (.../Bioinformatics for proteomics/Modeling/AT2F2_ATH.txt). The target structure should appear in the workspace together with the first template structure. You should see a linear structure that corresponds with the included raw target included.

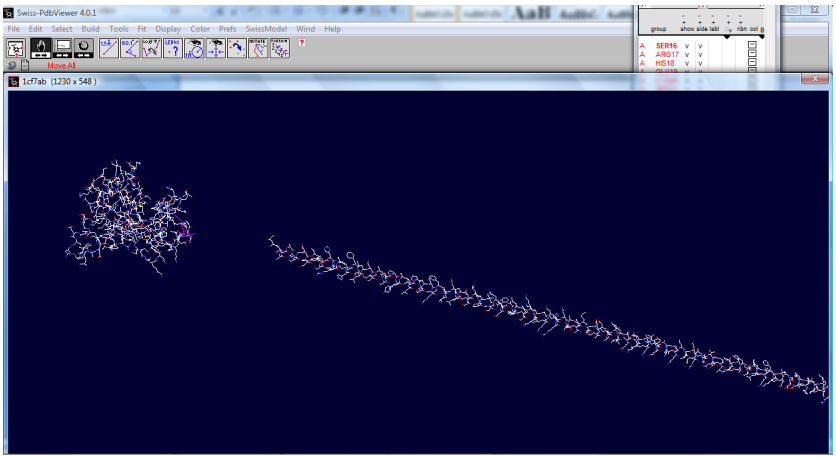


Fig. 8.14: Raw sequences appearance. The raw structure is indicated as a helical structure.

3. Center the view (Press "Home" key). A second row should appear in the Alignment Window (menu Window → Alignment) indicating the new included protein sequence. Now we need to superpose the protein structures and see how this kind of approach will work.

4. Make the initial target-template alignment. Execute: menu Fit → Fit Raw Sequence.

   Ignore the error message that says "No raw sequence". The long helix should disappear and instead the sequence should be now threaded onto the template.

   Note that the alignment in the Alignment Window has changed. Scroll from the N-terminus to the C-terminus to see the level of similarity. The sequence identity is also indicated with values to the right of the sequence labels in the window (first value corresponds to the overall identity, and second value corresponds to the identity of the current selected regions).

   Note that the alignment contains insertions and deletions. Insertions are not displayed in the Display Window, deletions are indicated as long artificial bonds connecting the flanking CA atoms.

   Save this alignment as AT2F2_ATH_raw_method.pdb by going to menu File → Save → Project (all layers). Close the layers. Go to File → Close all Layers. The alignment created

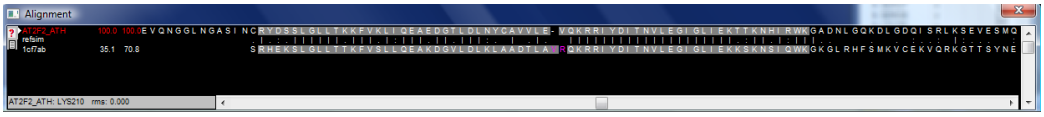


Fig. 8.15: Protein sequence alignment. The two entries are shown.

in the previous step might be much less accurate than the alignments using alignment data from fold Recognition servers. Let's see the difference (Koliński and Bujnicki, 2005). To do this, we need to use the data we obtained from the MetaServer (`www.genesilico.org`) analysis.

METHOD 2. *Using a model template from Metaserver.*

1. Download from the fold recognition MetaServer a crude model with the best scoring match to your template (1cf7_A in our case) as indicated by blastp. Do this by clicking the green button (model) located in the right side of the blastp box. Download the file and save the pdb file in your working directory. You will also find this file in our directory Modeling (18521-blastp-1cf7_A.pdb).

2. Open the model (18521-blastp-1cf7_A.pdb) in the DeepView program workspace (menu Open → PDB file...). You will likely be notified that some amino side chain atoms are missing. This is okay, as insertions are not present in this raw model. Thus, press OK. A log pop-up window will appear and show the missing atoms in the model. Have a look in the list and then close the pop-up window.

3. Go to menu SwissModel → Load Raw Sequence of amino acids... Load the target sequence file ATE2F2_ATH.fasta from the local folder (..Bioinformatics for Proteomics/Modeling).

4. The second row will appear in the Alignment Window.

5. Make the alignment of the target sequence to the template sequence. Go to menu Fit → Fit Raw Sequence. Ignore the unclear error message starting from "No raw sequence". The long helix should disappear and instead the sequence should be now threaded onto the model. Note that the alignment in the Alignment Window has changed.

6. Now go to menu File→ Open PDB File... and open the file 1cf7a.pdb. A third row should appear.

7. Go to menu Fit → Generate Structural Alignment (or press <Ctrl+G>). The layer corresponding to 1cf7ab should change its alignment in the Alignment Window.

8. Now select the layer corresponding to the model loaded first (18521-blastp-1cf7_A.pdb). The name in Alignment Window should turn red.

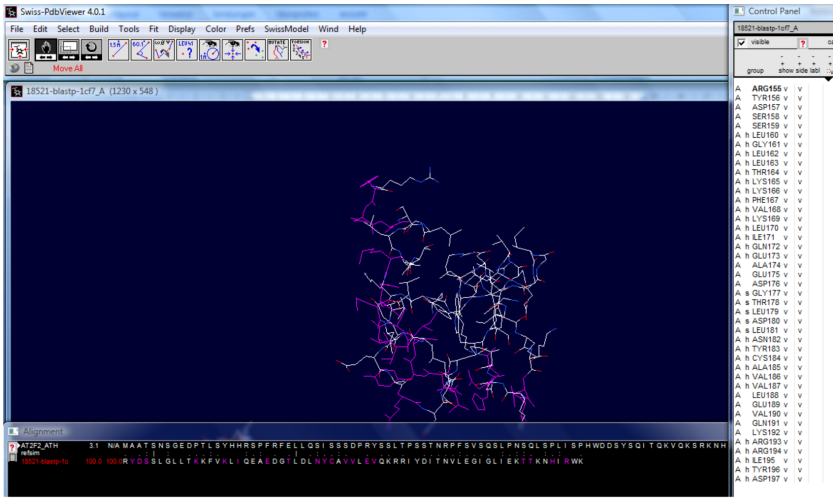


Fig. 8.16: Protein structural alignment. The target and template structures should appear aligned.

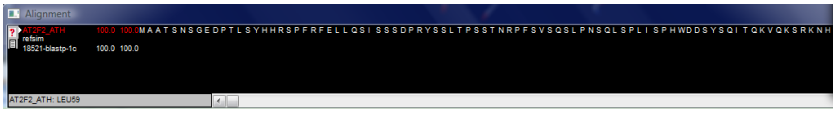


Fig. 8.17: Alignment window. The protein sequence alignment is changed.

9. Go to Menu File → Close. The layer corresponding to the model should disappear. If you can see two layers with the name ATE2F2_ATH now, close the Alignment Window and reopen it (there is a bug in the program).

Fig. 8.18: Second alignment window. Two sequences are now aligned.

10. Thread the raw sequence onto the new template. Select the first residue in the ATE2F2_ATH layer and move the shift the whole raw sequence back and forth using Ctrl-Space and Ctrl-Backspace.

11. Go to Menu File→ Save→ Project (all layers)... and save the file under the name ATE2F2_ATH_V1_project in your working directory. Congratulations! You have just created your first Swiss PDB Viewer project!.

Aligning of Structures

Automated Alignment

When you first load multiple structures in the workspace they will not be aligned; they will

show up in the orientation they were originally saved in. To ensure meaningful comparisons between structures, it is necessary to align them so that you can quickly see which parts of the structures correspond with each other.

It is possible to adjust the relative alignment of two structures by turning off the "Mov" attribute for one of them in the Layers Info window, and then moving the other one. This can be done, but there are other ways of doing structure alignment. For example, one way is to use the Tools available under the "Fit" menu. In particular, the "Iterative Magic Fit" tool is the one you should use for nearly all structural alignments. This tool will perform a sequence-level alignment of your structures to identify corresponding residue pairs. It then adjusts the alignment of the structures so that the maximum amount of residue pairs lie next to each other.

When you are aligning structures, the more atoms you try to align, the slower the alignment will be. For whole protein alignments, the structural alignment is basically just the alignment of the carbon backbones. Adding in side chains will slow things down and is not necessary.

Therefore:

- When aligning two normal proteins you only need to select "CA (carbon alpha)".
- For very small proteins you might be better off selecting "Backbone atoms only".
- For peptide fragments you might consider selecting "All atoms".

1- Apply the Iterative Magic Fit alignment between the two structures in your AT2F2_ATH_V1_project. Go to menu Fit → Iterative Magic Fit.

The alignment window should show a different sequence alignment.

The automatic alignment "Iterative Magic Fit" will be the process of choice for us on this tutorial, but other options are also available as mentioned in the following.

Manual Alignment

Sometimes an automated alignment isn't appropriate. This is usually because: (1) the sequences in your structures are very diverse, and the automated alignment cannot figure things out on its own, or (2) you want to bias your alignment around a particular set of residues (say the components of an active site) rather than align the whole protein. In these cases you can perform a manual alignment. To do this you must select the corresponding residues in the two structures. Your residues must be selected in corresponding pairs, and they must be the same residue in each structure. You need to select least three pairs of residues to perform this kind of alignment. The easiest way to make this kind of selection is with the alignment window. You can make this visible by selecting Wind → Alignment from the main menu. The alignment window shows you an alignment based on which residues are closest together in 3-dimentional space.

Selections in the alignment window work just like selections in the Control Panel. You can make discontinuous selections by holding down the Control Key.

Once you have selected your residue pairs you can start the alignment by selecting Fit → Fit Molecule (From Selection) from the main menu. You then go through the same process as for an iterative Magic Fit to align your molecules. Because you are using only a small number of

residues to calculate your alignment, you may want to use the "All atoms" option for performing the alignment.

Aligning more than two structures

The above methods work fine for aligning two structures, but they provide no option for aligning more than two. As of yet, there is no multiple alignment option for structures, but there is a simple way around this. If you want to align three or more structures you must do so by deciding that one of them should be the reference structure, and then aligning all of the other structures to it. Because structural conservation within a protein family is so high, this actually works well for creating a family alignment. You perform the individual alignments in the same way as previously described, but you must make sure that when you select the layers involved your reference structure always appears in the top box.

Lastly, once you have aligned all your structures, the sequence alignment in the Align window will probably be wrong. You can force it to update itself by selecting Fit → Generate Structural Alignment.

RMS Values

RMS values are often used to represent the degree of relatedness of two structures. The RMS value is the Root Mean Squared value for the average distance, in Angstroms, between corresponding atoms. The lower the RMS value, the more closely related the two structures are. By definition, a structure compared to itself has an RMS value of 0. Structures which are related will usually have an RMS value between 0 – 2Å. When you align two structures using either of the methods described above, DeepView will automatically calculate an RMS value for you. This is displayed in red text just below the buttons on the main toolbar.

When using the Iterative Magic Fit, the program will use all of the corresponding atoms it can find. For the manual alignment, it will only use the pairs of residues you specified.

Color Schemes to use with alignments

Once you have your structures aligned there are a couple of extra color schemes that become useful to you.

Layer

Coloring your molecules by layer simply assigns a different color to each open layer. This is a quick and easy way of being able to tell which residues come from which structure when they're all overlaid. The other advantage to this (and all other color schemes) is that the coloring applies not only to the view window, but also to the letters in the alignment window and the color blocks in the Control Panel.

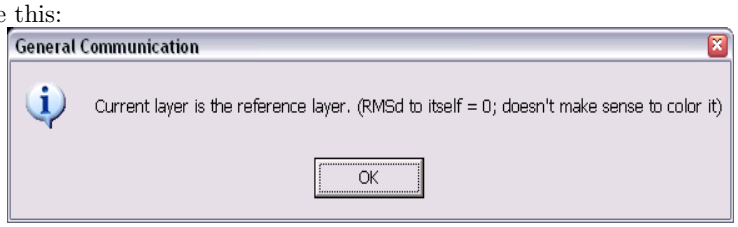
Alignment Diversity

This color scheme colors your models by the degree of conservation in their underlying sequence alignment. As with many of the other color schemes, it works on a cold → hot color series, where cold colors are the most conserved positions and hot colors are the most divergent. The alignment diversity color scheme will automatically apply itself to all the layers that are

currently open.

RMS The RMS color scheme colors your model by the amount of separation in 3D space between corresponding pairs of residues in your structure. It doesn't matter which residues are present in your sequences, this is a measure of how divergent the actual structures are. You will usually find that whereas two structures are quite divergent at a sequence level, they will usually be pretty well conserved at a structural level.

The RMS value for each residue is calculated by reference to your original reference structure. This means that if you have two structures aligned together, only one of them can be colored by RMS. If you try to apply an RMS color scheme to the reference layer, you will see a warning like this:



Usually it's a good idea when using the RMS color scheme to hide the reference layer (turn off the vis attribute in the Layers window), and just display the other layer(s) colored by RMS.

Saving Your Work If you have spent time creating a large family alignment, or setting up a specific view of a protein, then you don't want to lose that work when you shutdown DeepView. Fortunately, there are a couple of options you can use to save your work.

Saving Structures/Views

Single Layers

If you have a model that contains only one layer, then you can save this using:

File → Save → Current Layer

This will save the layer as a PDB file. The file created is a standard PDB file but with extra DeepView formatting codes in the header, which tell the program how the view was configured when you saved. DeepView will read these codes and re-create the view when you next open the file.

A PDB file should be able to run in other programs. You should be aware, however, that saving data from DeepView does not transform the underlying coordinates of the structure, so the view you have will not be preserved in anything other than DeepView. If you need to transform the raw data, then speak to the Bioinformatics group who can advise you on how to do this.

Multiple Layers If you have a structural alignment you wish to preserve then you can do this by using:

File → Save → Project (All layers)

This will create a single PDB file that contains all of the coordinates from all of the structures along with all of the view information to allow the scene to be recreated. You may find that if

you transfer this multiple PDB file to another structure display program that it does not open correctly. If you need to do this, please contact the Bioinformatics group, who may be able to help

What is lost when saving?

When you save your work in DeepView not everything is preserved by default. The things you will lose unless you take specific action are:

- Any molecular surfaces you had calculated.
- Any electrostatics potentials you had calculated.

Surfaces and electrostatic potentials can be saved, but they require you to perform an extra step.

Saving and Loading Surfaces and Electrostatics

To save a surface from the currently active layer use:

File→ Save→ Surface

To save an electrostatic potential from the currently active layer use:

File → Save → Electrostatic Potential

Please note that these only save from the current layer. If you have an alignment of 10 proteins and all have surfaces colored by electrostatic potential, then you will have to save 21 separate times.

To load a surface back onto a structure, you first need to reload the structure and make it the active layer. You can then reload the surface using:

File → Load Surface → SPDBV

Likewise for Electrostatic Potentials you would use:

File → Load Electrostatic Potential → SPDBV.

8.4. Comparative data analysis: Image analysis and identification of proteins

8.4.1. Introduction

Successful analysis of large-scale data in proteomic approaches is increasingly dependent on the advances on bioinformatic tools and statistics for comparative data analysis (Stead *et al.*, 2008). In the gel-based proteomic approach, the use of programs for image analysis is extremely necessary. In those studies the proteins are separated in the bi-dimensional gels (2D gels) and the images of those gels are compared to determine differences in the protein abundance (Görg *et al.*, 2009). In non-gel based proteomic approaches the proteins are usually from different samples and are labeled with different "tags" (isotopes with different masses) and combined. Those proteins are separated by liquid chromatography and identified with mass spectrometers. Specific programs that are able to extract the values of mass/charge (m/z) and charge state of the RAW data are used to process big amounts of data simultaneously. From the RAW data it is also possible to extract the intensities of each peak and compare them in order to find the different peak intensities among the isotopic pairs, resulting in a relative quantification of proteins. The identification of proteins in the proteomic experiments is a bioinformatic task per se. Many efforts are still needed to improve the quality and sensitivity of the protein identification methods (Mueller *et al.*, 2008).

First Instructions

On the present session we will learn how to do basic analysis of 2D-gel images and identify proteins by Peptide mass fingerprint and Tandem MS/MS. The files needed for this session are in a local directory (../Bioinformatics/Protein identification/).

Image analysis: The base for comparative gel-based proteome research

The results of a comparative gel-based proteome study are directly related to the analysis of the gel images obtained experimentally. This makes such a task more laborious and time consuming. Usually the 2D gels used in comparative proteome studies are stained with a chemical named Comassie B. Blue and scanned in special densitometers. The objective of this is to capture the optical density of the protein spots and compare this data with the different gels from different samples. Quantification of the spots is one of the critical factors for a successfully comparative proteome analysis. The experience of the operator can have an important consequence in the final result.

There are many available programs for 2D-image analysis, but few of them are free of charge. We will shortly practice the main tools of one commercial program named Melanie. This program was the first program dedicated for 2D-image analysis and is now commercialized. We will see some features of this program and learn how to use the basic tools for comparative analysis of 2D gels.

Design of the comparative analysis

1. Open the program Melanie in the icon in your desktop. You will see the main tool bar in the upper part of the window and a workspace window in the left side.
2. In the workspace window, click on the small arrow in the folder icon and select "New". This will open a new small window named "New Project". Include a name for your project "Curso Bioinfo" and click OK.
3. Go to menu File → Open. A new window will appear for you to select the gel images you have in your local directory (...Bioinformatics for proteomics/Protein identification/Gels). Select all six gels that are inside of this folder.
4. See that the gels appeared in the workspace window inside "Image Pool" folder.

   Now we have to create the design of the analysis.

5. In the workspace window, select the folder "Curso Bioinfo" and right click on it and select "Create Match Set". In our analysis we will create two Match sets, one of them named "T1" (treatment 1) and other named "T2" (treatment2).
6. After creating the "Match Set" folders, copy the gels from the folder "Image Pool" to the respective "T1" and "T2" Match folders.
7. Double click in the Match folder "T1" to open the T1 gels and do the same for the T2 gels.

8. Go to menu View→Gels→Adjust contrast. In the y axis of the graphic of intensity, reduce the contrast in 1 point moving the arrow. Click "Apply". Close this contrast window.

9. Go to menu View→Global→Show Gel Names.

10. Go to menu View→Sheet→Align Images.

11. Go to menu View→Spots→Outlined.

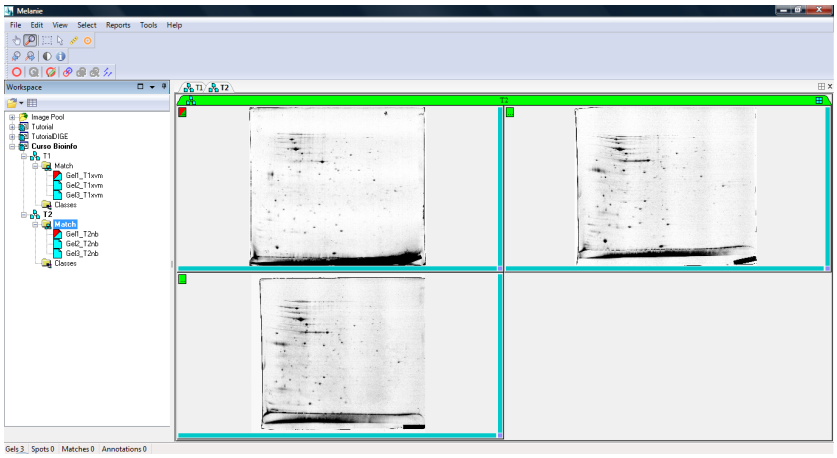


Fig. 8.19: Melanie. Workspace and loading gels.

12. Select a small region in one gel by using the tool "Region" located in the main tool bar.

Fig. 8.20: Main tool bar. Control Panel quick tools. The select region tool is highlighted.

13. Go to menu Edit→Spots→Detect.

14. Go to the small icon on the right side of the gel images, left click on the icon and select "Stackled". You should see the selected area of the gel zoomed.

    You can see that many artifacts are detected as protein spots. We can try to improve the spot detection by adjusting the parameters shown in the "Detect Spots" window.

15. Go to menu View → Global → Cursor Information Window. A cursor information window should open.

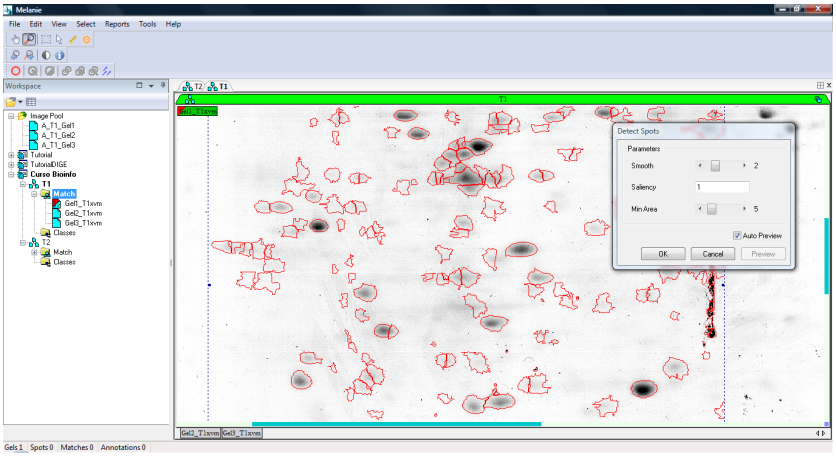


Fig. 8.21: Spot detection. How to optimize the automatic spot detection.

16. Move the cursor over a few spots that you consider not to be a protein spot and see the "Saliency" value of them in the cursor window. Apply a value of saliency in the cursor window bigger than the artifact's saliency. If you apply a Saliency value of 2.

17. Select now the total area of all gels and select all gel images inside the "Match" folders.

18. Go to menu Edit → Spot → Detect. The program will detect the spots of all gels from one treatment. In the Detect Spots window, click OK. Repeat this process for the gels of the second treatment gels.

    Now we will add some Landmarks for performing the gel matching.

19. Click Landmark in the toolbar.

20. Position the cursor over a known, well-defined spot in the reference gel and click. A "Validated" landmark symbol (bold orange circle) appears on the spot.

21. In the other images, drag the "Non-validated" symbols (green circle with orange plus sign) onto the corresponding spots.

22. Do this for two landmarks.

23. Select the Match sets T1 and T2 using the Ctrl+, right click in the T1 and select "Merge MatchSet". Give a new name to this Match set and click OK.

24. Open the new Match Set and see all the gels in the same panel.

25. Select all gels by clicking on their names. Go to tool bar and click on the icon "Match Gels". Select which gels will be compared in the match process.

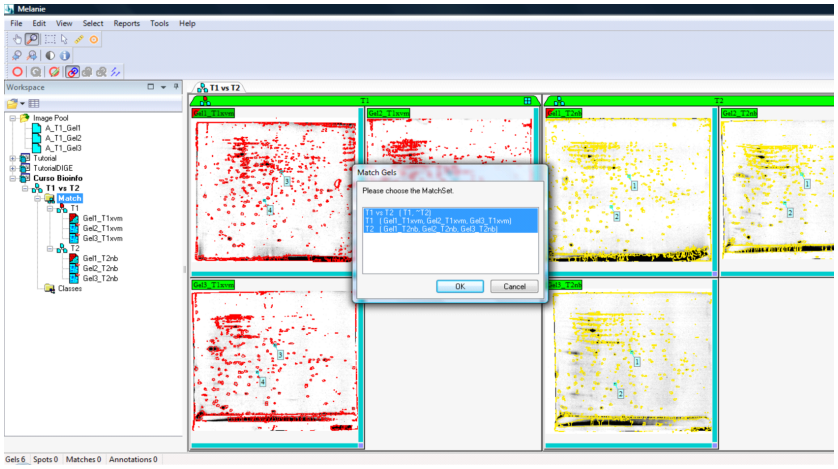


Fig. 8.22: Gels matching. The spot intensities are compared between gels.

When a gel is moved, the vectors become longer. Select Move in the toolbar and double click in the gel to re-center all the images, including the reference, to the same position and therefore minimize the match vectors. A blue upside down triangle on a spot indicates that the spot was matched to one or several spots in other gels, for which no corresponding spot in the global match reference was found. A spot with a triangle means that the corresponding position in the reference lies outside the visible area.

26. In the "Match" folder, select all gels to be added to a new class. The gels may be part of different sub Match sets but must belong to the same Match hierarchy.

27. Right click and select "Add in Class".

28. Enter a name and click OK.

29. Repeat these steps until all classes are created.

30. Hold the Shift or Ctrl key to select several classes.

31. To open the images in a classes sheet, right-click and select Display.

32. Go to menu Tools→ Options→ Quantification→ OK.

33. Go to menu Reports→ Analyze Classes→ Table.

    The table containing quantitative data of the protein spots should open.

    It is now possible to perform some different statistics.

34. Open the drop down list on the Class Analysis Table and select "Ratio".

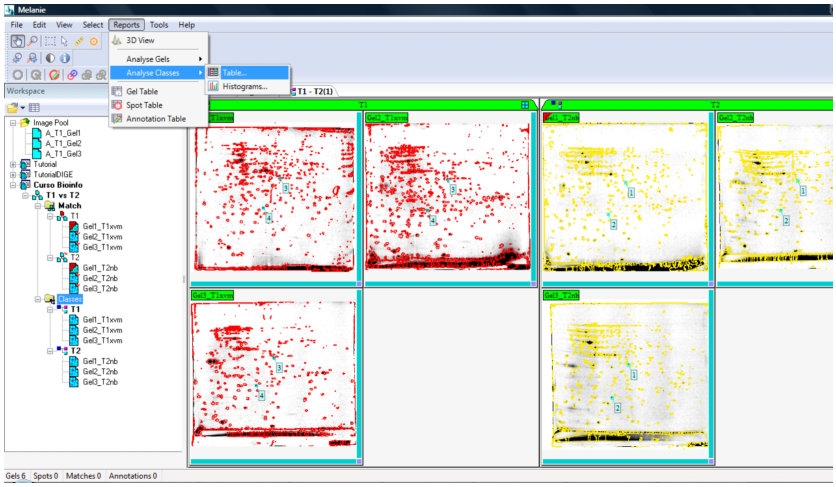


Fig. 8.23: Quantitative analysis. How to open the analyses table.

35. Click on the icon on the left side of the drop down list and select all items. You now have a table containing all the statistics possible to be performed in this program.

36. Save the Class Analysis table. Go to the second icon on the left side of the table and click on the drop down arrow. Select "Save" and choose your local directory.

37. You can also visualize data analysis by means of histograms. Go to menu Reports→ Analyze Classes→ Histograms. You will see the Histograms corresponding to each Match ID. The histograms contain the quantification of the volume percentage of the spots in the Class T1 (left side vertical bars) and T2 (right side vertical bars).

38. Go to the second icon on the left side of the Histograms panel and click on the drop down arrow. Select "Save" and choose your local directory.

Identification of proteins

The Mass spectrometry is an analytical method used to determine the masses of chemical compounds. There is equipment able to determine the mass of intact proteins and peptides, and there is equipment able to determine the amino acid sequence of peptides. In almost all cases, these tools are used to reveal the identity of a certain protein or groups of proteins. However, it is also possible to recognize post-translational modifications of proteins.

There are many programs available for identification of proteins. Some of these are online versions (e.g. Mascot, OMSSA and ProteinProspector) and others just run locally (e.g. TPP, X!tandem and Sequest).

For identification of proteins in large-scale experiments usually the pipeline for data processing includes file conversion, database searches, data exporting and database development. Such

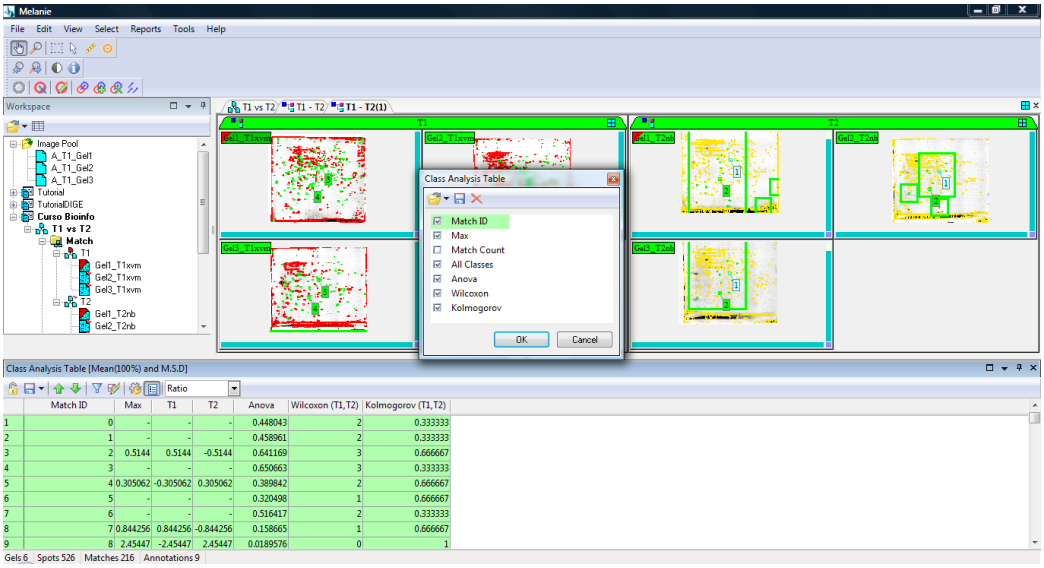


Fig. 8.24: Results. The results can be viewed in a table or by histograms.

analysis needs the programs to run locally and usually involves the administration of a server for computing the search on locally installed databases.

In our tutorial, we will learn how to identify proteins by two methods using Mascot search engine: peptide mass fingerprint (PMF) and tandem mass spectrometry (tandem MS/MS).

Proteins of a green algae named *Chlamydomonas reinhardtii* were analyzed by MALDI-ToF (for PMF identification) and by LC MS/MS (for tandem MS identification) resulting in the data files used in this tutorial.

Peptide Mass Fingerprint (PMF)

1. Open the website of Matrix Science (`http://www.matrixscience.com/`)
2. Click in "Mascot"

   A new page should open with three options of protein identification.

3. Select "Peptide Mass Fingerprint". An empty form will be opened. Fill in the form by adding your name and e-mail address in the indicated area. Include the additional information as follow:
   Search title: SampleH
   Database: MSDB

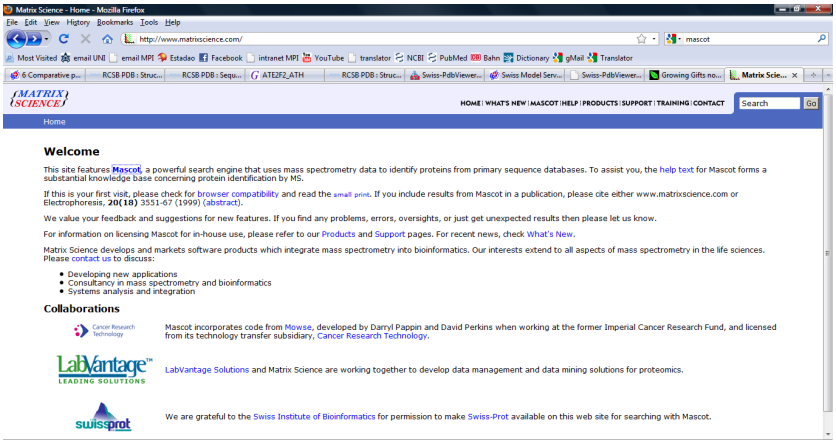


Fig. 8.25: Matrix scicence. Mascot program can run on-line for small-scale proteomic studies.

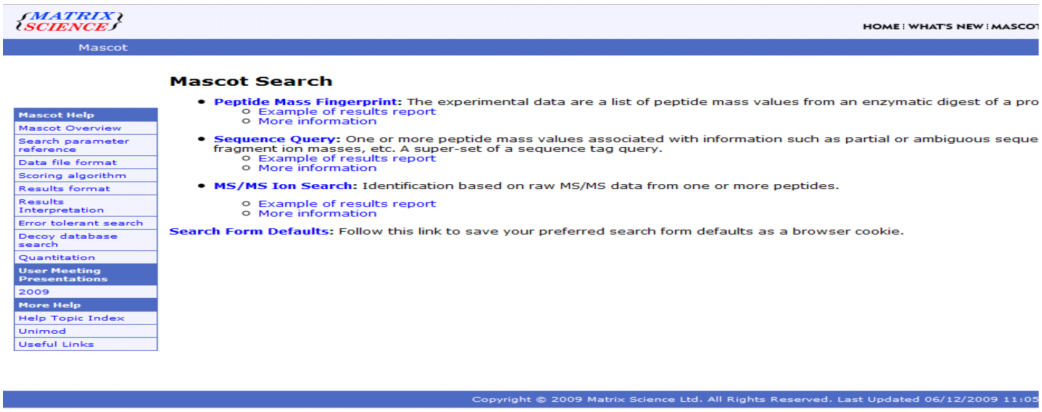


Fig. 8.26: Mascot. Options for performing protein identification.

Taxonomy: All entries
Enzyme: Trypsin
Allow up to : 1 missed cleavages
Do not select any modification and do not include any protein mass.
Mass values: MH+ active
Monoisotopic: active
Data file: Go to "Browse" and open the file containing the peak list of the SampleH (../Bioinformatics for proteomics/Protein identification/PMF/SampleH_peaklist.txt)

4. Click on "Start Search".

   A results page should open. You can scroll down the page and see the detailed information of the protein hits.

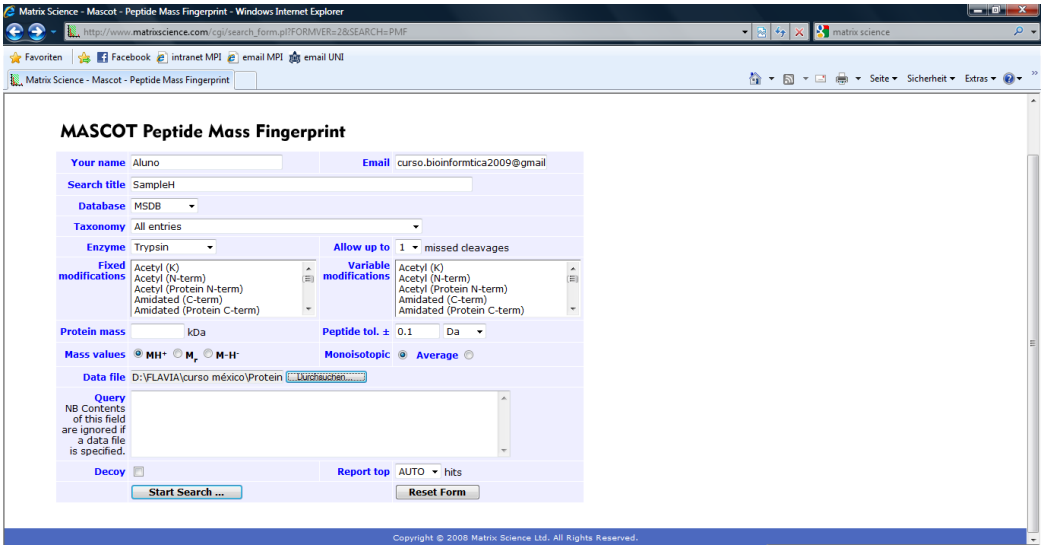


Fig. 8.27: PMF parameters. Many search parameters are available on-line.

Now we will have a look at the description of the results and export the data.

1. Go to the first protein hit (H4_CHLRE) in blue and look at the Score value. For validating the protein identification, ask yourself the following questions:

   a) Is it the protein Score greater than the Mowse Score?

   b) From which specie is this protein?

   c) What is the protein coverage?

   Export the results

2. Go to Format As → Export Search Results. Select a local folder for saving your files.

   A new page will open for you to select which parameters of the results you want to save. Most of them are already preselected. Many of them are not available for online versions of the search. They will only appear when you have a local copy of the program that runs in a Server.

3. Go to "Export format" and select "CSV" option. Scroll down the page and click on the button "Export". A window will appear for you to save the data into a local directory.

Congratulations! Your first protein was identified by peptide mass fingerprinting.

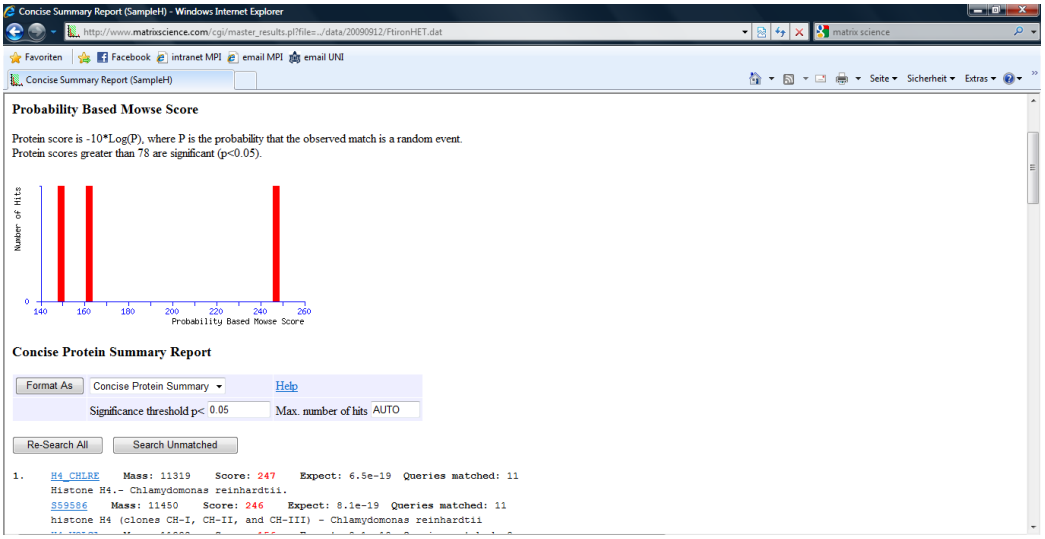


Fig. 8.28: Results PMF. A Protein summary is shown.

Now you can imagine how laborious this would be to identify thousands of proteins by PMF. Fortunately, this is not the case for high throughput analysis. Now we will see how to identify many proteins at the same time using MS/MS.

Identification of Proteins by tandem MS/MS

1. Go back to the home page of the website of Matrix Science where we could see the three search options (`http://www.matrixscience.com/search_form_select.html`).
2. Select "MS/MS ion search"
3. Fill the search form as indicated below:

   Include your name and e-mail address in the respective areas.

   Search title: SampleA_fraction19
   Database: NCBInr (National center of Biotechnology Information non-redundant)
   Taxonomy: All entries
   Enzyme: Trypsin
   Allow up to: 1 missed cleavages
   Fixed modifications: Carbamydomethyl (C)
   Variable modifications: Oxidation (M)
   Quantitation: None
   Peptide tol: 10 ppm
   MS/MS tol: 0.8Da

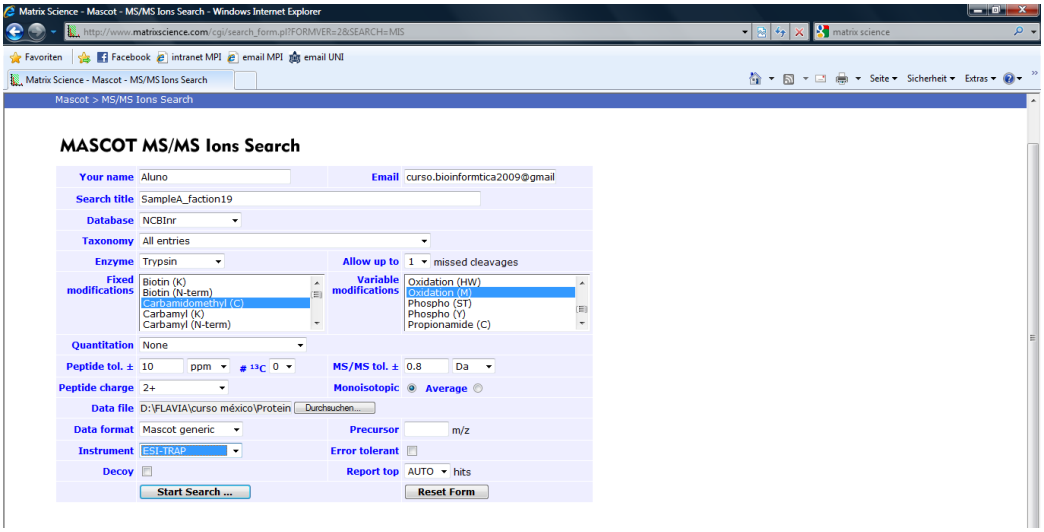


Fig. 8.29: Tandem MS/MS search. Specific parameters are included for tandem MS/MS data.

Peptide charge: 2+
Monoisotopic: active
Data format: Mascot generic
Instrument: ESI-TRAP
Data file: Go to "Browse" and open the file containing the peak list of the SampleA_fraction19 in Mascot generic file format (extension.mgf) (../Bioinformatics for proteomics/Protein identification/ShotgunProteomics/SampleA_fraction19.mgf)

4. Click on "Start Search".

   The data will be processed and the results page should open.

5. Go to "Export format" and select "CSV" option. Scroll down the page and click on the button "Export" A window will appear for you to save the data into a local directory.

6. Open the result file in Excel and see what kind of data you have for the extraction of the identity of the detected proteins.

Usually in large-scale data analyses it is not possible to manually check the identification of all proteins. Here we analyzed just a very small fragment of a dataset of a proteomic investigation. It is not possible to deal with large amounts of data in the online versions of the Mascot software.

For working with large datasets you can use several pipelines available, such as Trans Proteomic Pipeline (TPP) or develop your own pipeline for data analysis.

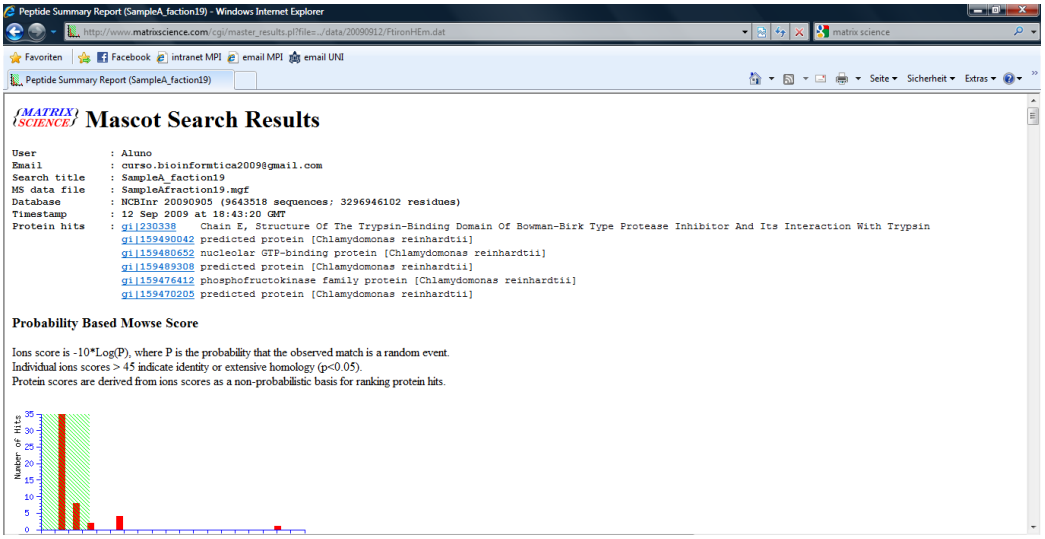


Fig. 8.30: Results of tandem MS/MS. The number of proteins identified usually is very big.

Bibliografía

Abascal, F., R. Zardoya, and D. Posada, 2005 ProtTest: selection of best-fit models of protein evolution. Bioinformatics **21**: 2104–2105.

Anfinsen, C. B., E. Haber, M. Sela, and F. H. White, 1961 THE KINETICS OF FORMATION OF NATIVE RIBONUCLEASE DURING OXIDATION OF THE REDUCED POLYPEPTIDE CHAIN. Proceedings of the National Academy of Sciences of the United States of America **47**: 1309–1314.

Assenov, Y., F. Ramirez, S.-E. Schelhorn, T. Lengauer, and M. Albrecht, 2008 Computing topological parameters of biological networks. Bioinformatics **24**: 282–284.

Bourne, P. E., 2004 The future of bioinformatics. In *2nd Asia-Pacific Bioinformatics Conference (APBC2004)*.

Clark, R. M., G. Schweikert, C. Toomajian, S. Ossowski, G. Zeller, *et al.*, 2007 Common sequence polymorphisms shaping genetic diversity in arabidopsis thaliana. Science **317**: 338–342.

Cline, M. S., M. Smoot, E. Cerami, A. Kuchinsky, N. Landys, *et al.*, 2007 Integration of biological networks and gene expression data using cytoscape. Nat Protoc **2**: 2366–2382.

Dobzhansky, T., 1973 Nothing in biology makes sense except in the light of evolution. The American Biology Teacher **35**: 125–129.

Eddy, S. R., 2004a What is dynamic programming? Nat Biotechnol **22**: 909–10.

Eddy, S. R., 2004b Where did the BLOSUM62 alignment score matrix come from? Nat Biotechnol **22**: 1035–6.

Emig, D., M. S. Cline, T. Lengauer, and M. Albrecht, 2008 Integrating expression data with domain interaction networks. Bioinformatics **24**: 2546–2548.

Fitch, W. M., 2000 Homology a personal view on some of the problems. Trends Genet **16**: 227–231.

Gomez-Porras, J. L., D. M. Riaño Pachón, I. Dreyer, J. E. Mayer, and B. Mueller-Roeber, 2007 Genome-wide analysis of ABA-responsive elements ABRE and CE3 reveals divergent patterns in Arabidopsis and rice. BMC Genomics **8**: 260.

Görg, A., O. Drews, C. Lück, F. Weiland, and W. Weiss, 2009 2-DE with IPGs. Electrophoresis **30**: 122–132.

Haider, S., B. Ballester, D. Smedley, J. Zhang, P. Rice, *et al.*, 2009 Biomart central portal–unified access to biological data. Nucleic Acids Res **37**: W23–W27.

Henikoff, S., and J. G. Henikoff, 1992 Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci U S A **89**: 10915–10919.

Ideker, T., V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, *et al.*, 2001 Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. Science **292**: 929–934.

Killcoyne, S., G. W. Carter, J. Smith, and J. Boyle, 2009 Cytoscape: a community-based framework for network modeling. Methods Mol Biol **563**: 219–239.

Koliński, A., and J. M. M. Bujnicki, 2005 Generalized protein structure prediction based on combination of fold-recognition with de novo folding and evaluation of models. Proteins .

Lemey, P., M. Salemi, and A.-M. Vandamme, editors, 2009 *The Phylogenetic Handbook*. Cambridge University Press.

Leonard, S. A., T. G. Littlejohn, and A. D. Baxevanis, 2007 Common file formats. Curr Protoc Bioinformatics **Appendix 1**: Appendix 1B.

Maere, S., K. Heymans, and M. Kuiper, 2005 Bingo: a cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. Bioinformatics **21**: 3448–3449.

Mizrachi, I. K., 2008 Managing sequence data. Methods Mol Biol **452**: 3–27.

Mueller, L.Ñ., M.-Y. Brusniak, D. R. Mani, and R. Aebersold, 2008 An assessment of software solutions for the analysis of mass spectrometry based quantitative proteomics data. Journal of Proteome Research **7**: 51–61.

Notredame, C., 2007 Recent evolutions of multiple sequence alignment algorithms. PLoS Comput Biol **3**: e123.

Posada, D., 2006 Modeltest server: a web-based tool for the statistical selection of models of nucleotide substitution online. Nucleic Acids Res **34**: W700–W703.

Shannon, P., A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, *et al.*, 2003 Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res **13**: 2498–2504.

Stead, D. A., N. W. Paton, P. Missier, S. M. Embury, C. Hedeler, *et al.*, 2008 Information quality in proteomics. Briefings in Bioinformatics **9**: 174–188.

Stein, L. D., 2008 Bioinformatics: alive and kicking. Genome Biol **9**: 114.

Wilkinson, M., J. O. McInerney, R. P. Hirt, P. G. Foster, and T. M. Embley, 2007 Of clades and clans: terms for phylogenetic relationships in unrooted trees. Trends Ecol Evol **22**: 114–115.

Apéndices

PROCESSES

`^c <ctrl>-c` kills (definitely stops) current job

`^z <ctrl>-z` suspends the current job. This can either be moved to the background or resumed in the foreground by using `bg` or `fg`

`bg` moves the current process to the background

`fg` moves a process to the foreground. (If there is more than one suspended job, use `jobs` to decide which you want to `fg`)

`fg 2` moves process number 2, as listed by jobs, to the foreground

`jobs` lists background and suspended processes (created with `bg` or `^z`)

`jobs -l` ("el" not one) includes the pid (process id number)

`ps` lists all your processes

`kill` stops a process (use `ps` or `jobs` to find your processes)

`kill 2986`
kills off the process with pid *2986*

MISCELLANEOUS

`finger` tells you who is logged on (see also `w`)

`w` shows information about logged in users

`who` produces similar result (see finger)

`tar` create (or extract) a tarball from (to) a list of files

`tar -cvf tarball.tar subdir/*`

`tar -xvf tarball.tar`
the option `-z` compacts the files by `gzip`

`wc` word count

`wc long.file`
prints the number of lines, words and characters in *long.file*. Options include `-l` to count lines only, and `-c` to count characters only

`ln` create a link or an alias for a file

`ln -s subdir/orig.file alias.file`

`history` displays last several commands used

`!!` re-executes the last command

`!51` executes command 51 in the history list use also <`up`> - and <`down`> - arrows to navigate in the history

`date` displays current date and time

`passwd` invokes a password changing program

`exit` leaves the current shell (same as `^d` or `<ctrl>-d`) usually = logout

GRAPHIC DISPLAY

To display graphics, most Unix require the configuration of the X-Window server.

Commands on your local computer:

`xhost` set the list of allowed X-Window clients

`xhost +`
The "`+`" allows any remote computer to display on your local display

`ifconfig` gives information about the network configuration (e.g., the current IP_address, usually similar to 123.145.167.189)

Commands on the remote computer:

`setenv` set up an environment variable (tc-shell)

`setenv DISPLAY IP_address:0.0`
required to tell the remote computer where it should display its graphics

`xclock` starts a graphic clock (e.g., used to test the X-Window server or to get the current time... ;-)

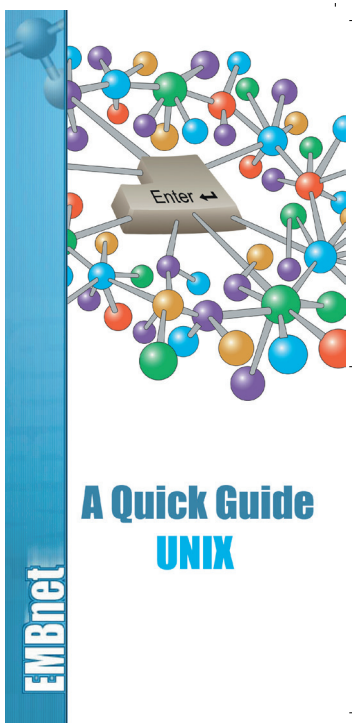
This document was originally written and designed by Aoife McLysaght and Andrew Lloyd© from the Irish EMBnet node, and modified by Laurent Falquet from the Swiss EMBnet node and distributed by the Publications Committee of EMBnet.

EMBnet - European Molecular Biology network - is a network of bioinformatics support centres situated primarily in Europe. Most countries have a national node which can provide training courses and other forms of help for users of bioinformatics software.
Further information about UNIX is available from your national node. You can find contact information about your national node from the EMBnet web site:

http://www.embnet.org/

If you have found this publication useful, please let us know.
If you have ideas for similar documents we'd like to hear from you: emb-pr@embnet.org

A Quick Guide To UNIX
Revised edition 2003



A Quick Guide
UNIX

A Quick Guide To UNIX

This is an introduction to the UNIX operating system. Unix may seem idiosyncratic, even impenetrable, to begin with but it has the virtue of minimising the number of keystrokes and so speeding up your access to the computer.
The commands listed here are common to different operating systems and shells. They include some of the most useful and frequently used commands in UNIX. The power and utility of most UNIX commands can be enhanced with switches or options preceded by a "**-**" sign.
More information on the options, the effects and how to use the commands is available by using the **man** command:

man gives manual information on a topic
man grep
displays the manual page about grep
apropos lists all the man(ual) entries relating to a topic (same as **man -k**)
apropos print

Another useful source of information is the on-line EMBnet tutorial which includes a page on UNIX

http://www.dk.embnet.org/Embnetut/Universl/unixcmds.html
or equally
http://www.uk.embnet.org/Embnetut/Universl/unixcmds.html

The general format of this document is that anything in **bold** is a command you can enter. Anything in ***italic*** is a fake file or directory name you must change according to yours. Anything preceded by a hyphen "**-**" is an option which will modify the effects of a command. A general description of each command is followed by one or several examples of its use.

FILES

ls lists files in a directory
ls -alF
lists **-a** all files in **-l** long format **-F** identifies directories **/**, executable files *****, and symbolic links **@**, in the current directory

cat concatenates and displays files
cat *my.file*
displays *my.file* on the screen

chmod modifies the read (**r**), write and delete (**w**), and execute (**x**) permissions of specified files and the search permissions of specified directories. The permission can be set for user (**u**), group (**g**) or other (**o**)
chmod go-w *my.file*
stops (**-**) anyone else (**go**) changing or deleting (**w**) *my.file*
chmod g+rwx *my.file*
allows (**+**) anyone of my group (**g**) reading, changing, deleting or executing (**rwx**) *my.file*

cp copies files
cp *orig.file copy.file*
cp *orig.file subdir/new.file*
copies *orig.file* to *new.file* in *subdir* directory
cp *subdir/orig.file* .
copies *orig.file* from *subdir* to the current directory (**.**) without changing its name

mv moves/renames a file (or directory)
mv *oldname newname*
mv *my.file subdir/my.file*
a move (**mv**) is equivalent to a copy (**cp**) followed by a remove (**rm**)

rm removes/deletes a file.
rm *oldfile*
rm -i **.file*
option **-i** (interactive) advised if wildcards (*****) in use

diff compares two files and prints how they differ
diff *file1 file2*
prints differences to screen options include **-b** to ignore differences in blank space, and **-i** to ignore case

find searches the directory tree for a file
find . -name *lostfile* -print
will search your current directory (**.**) (and any subdirectories) for *lostfile*

grep searches a file for a string
grep *word my.file*
grep "two words" my.file
options include **-i** to ignore case and **-n** to print line numbers

vi simple screen oriented text editor

pico simple display oriented text editor
pico *myfile.txt*

head prints the first few (default = 10) lines of a file
head *oddfile*
head -20 *oddfile*
displays first twenty lines of *oddfile*

tail displays last few lines of a file (see head)

more displays a file one screenful at a time
more *longfile*
hit <**spacebar**> to see the next screen
Note: some people prefer **less**

OUTPUT REDIRECTION

> redirects output of a command to a file
diff *file1 file2* > *new.file*
puts differences into *new.file*
cat *one.file two.file* > *both.file*
writes the output of the cat command into *both.file* (overwrites *both.file*)

>> appends a file to the bottom of another
cat *three.file* >> *both.file*
appends *three.file* to the bottom of *both.file*

| "pipe" - uses the output of the first command as the input of the second
grep *string my.file* | wc -l
finds how many lines on which *"string"* occurs (see **grep** and **wc**)

DIRECTORIES

cd changes current directory
cd */etc*
go to */etc* directory
cd ..
go up one level in directory tree
cd *../subdir2*
go "sideways" to *subdir2*

mkdir creates a new subdirectory
mkdir *subdir*

rmdir removes a directory - you must delete all the files in it first
rmdir *subdir*

pwd print working directory, tells your current location (path)

e.g `seqret "embl:hs*"`

A part of the sequence can be specified by adding the range:

e.g `seqret "embl:hsfau[1:57]"`

The last 100 bases of a sequence can be specified by a negative start:

e.g `seqret "embl:hsfau[-100:]"`

List Files

A list file contains a list of USAs (one per line). The list file input is `@listfile`. A list file may be read in wherever a program can read multiple sequences. Blank lines and USAs starting with a '#' character are ignored. There is no limit on different sequence formats within one list file.

Format Conversion

The format of an output sequence file can be specified. `seqret` can read in sequences in one format and write them in the other format, for example to convert a sequence to GCG format:

`seqret` *in.seq* `gcg::`*out.seq*

The command line and parameters

EMBOSS programs are designed to be run from the command-line, as well as within scripts. To customise their behaviour, each has a distinct set of parameters, also known as options or flags.

There are 3 classes of parameters: *standard*, *additional*, *advanced*. Information on allowable flags for each program is given in the help files.

If values for *standard* (mandatory) parameters are not specified, the programs will prompt for them.

If *additional* (optional) parameters are missed out, default values will be used unless you put `options` (or `opt`) on the command line.

EMBOSS programs never prompt for *advanced* parameters; these must be explicitly specified. They are defined in the program documentation.

General qualifiers

These can be used with any program:

`-auto`	Turns off prompts and descriptions. Used when in running programs scripts
`-stdout`	Writes to standard output (screen) by default
`-filter`	Reads from standard input (keyboard), writes to standard output (screen) by default
`-options`	Prompts for all required and additional values
`-debug`	Writes debug output to the file *programname.dbg*
`-help`	Reports command line options. Or help verbose for more information on associated and general qualifiers
`-warning`	Reports warnings
`-error`	Reports errors
`-fatal`	Reports fatal errors
`-die`	Reports deaths

Each of these can be prefixed with "no" to negate the action.

e.g. `-nowarning`

`-sbegin`	States the first position of the sequence
`-send`	States the final position of the sequence

Some major programs

EMBOSS currently offers approximately 200 applications Use `wossname` to see them all together with below a selection of interesting tools:

TOOLS (examples)

`seqret`	Reads and writes (returns) sequences
`est2genome`	Aligns EST and genomic DNA sequences
`needle`	Needleman-Wunsch global alignment
`water`	Smith-Waterman local alignment
`dotmatcher`	Displays a thresholded dotplot of two sequences
`remap`	Displays a sequence with restriction cut sites, translation etc
`prettyplot`	Displays aligned sequences, with colouring and boxing
`extractseq`	Extracts regions from a sequence
`revseq`	Reverses and complements a sequence
`plotorf`	Plots potential open reading frames
and many other	

UTILS MISC

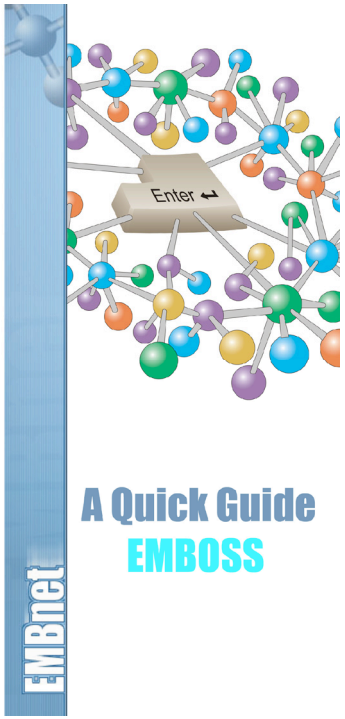
`embossdata`	Finds or fetches the data files read in by the EMBOSS programs
`embossversion`	Writes the current EMBOSS version number

This document was written and designed by Lisa Mullan from the UK EMBnet node and being distributed by P&PR Publications Committee of EMBnet.

EMBnet - European Molecular Biology Network - is a bioinformatics support network of bioinformatics support centers situated primarily in Europe. Most countries have a national node which can provide training courses and other forms of help for users of bioinformatics software.

You can find information about your national node from the EMBnet site:

http://www.embnet.org/



A Quick Guide
EMBOSS

A Quick Guide To EMBOSS

http://www.emboss.org

This is a Quick reference Guide for EMBOSS version 2.8.0

Rice, P. Longden, I. and Bleasby, A. (2000) "EMBOSS: The European Molecular Biology Open Software Suite" *Trends in Genetics* **16**(6):276-277.

Introduction

EMBOSS (European Molecular Biology Open Software Suite) is a freely available suite of programs and libraries for sequence analysis. It incorporates many tools originating from the EGCG package created in 1988. All EMBOSS programs are designed to run on a UNIX command-line or behind graphical interfaces (*e.g.*, Jemboss, wEMBOSS).

Obtaining EMBOSS

To install EMBOSS: download the current version from `ftp://ftp.uk.embnet.org/pub/EMBOSS/EMBOSS-2.8.0.tar.gz`, then follow the instructions at:
`http://www.rfcgr.mrc.ac.uk/software/EMBOSS/download.html`

Graphical User Interfaces

There are a number of graphical interfaces to EMBOSS:
`http://www.rfcgr.mrc.ac.uk/Software/EMBOSS/interfaces.html`

Jemboss is a java interface and is distributed with EMBOSS. If you are installing with the Jemboss interface you should use the installation script in the `EMBOSS-x.x.x/jemboss/utils` directory. Instructions for Jemboss installation are given at:
`http://www.rfcgr.mrc.ac.uk/Software/EMBOSS/Jemboss`

Support and Mailing lists

The mailing list `embcss@embnet.org` is used for discussions of user problems. To subscribe to this list, send a mail to `majordomo@embnet.org` with the message text: *subscribe emboss*. The mailing list archive is:
`http://www.rfcgr.mrc.ac.uk/Emboss/HYPERMAIL/emboss`

Please send bug reports to `emboss-bug@embnet.org`

Help on a program

A program can be found using a keyword search of the description of all the programs by running the EMBOSS application `wossname`.

`wossname` *keyword*	displays list of all programs with keyword in description
`wossname -alphabetic -auto`	displays a list of all programs
programname `-help`	gives the available parameters for the *programname*
`tfm` *programname*	displays the documentation of *programname*

`Documentation is also given online at:`
`http://www.rfcgr.mrc.ac.uk/Software/EMBOSS/Apps`

Sequence formats

Sequences are stored in databases or in files as simple text. EMBOSS does not support sequences in word-processor files! The default sequence file format is **fasta**. This format has an initial title line consisting of a ">" followed by the sequence description on the first line. The second and subsequent lines contain the sequence, *e.g.:*

```
>fau Human FAU gene fragment
GACCGGCCAGGAAACGGCATGTAGCCTCACTGGAGGGCATTGCCCCGGA
AGATCAAGT
```

EMBOSS currently supports 42 formats, including: **Clustal, EMBL, GCG, Genbank, PIR, MSF, Phylip, SwissProt, Text (raw).**
The default output can be altered for all programs by an environment setting:
`setenv EMBOSS_OUTFORMAT format`

Alignment Formats

Several formats have been written or adopted for EMBOSS output.

Multiple Alignment

`simple`	Displays names, positions and sequences, markup line underneath [default]
`fasta`	Standard fasta display. Gaps displayed as "-"
`msf`	Standard MSF format. Gaps displayed as "." for intrinsic and for terminal ones
`srs`	Similar to simple. No markup line
`trace`	Verbose form for de-bugging

Pairwise Alignment

`pair`	Simple format for pairwise output [default]
`markx`	Standard output from FASTA program suite
`srspair`	Similar to pair format
`score`	Score output. No sequence display

Any program derived from Bill Pearson FASTA suite of programs has a markx default format.

`-aformat`	Alters output format
`-awidth`	Displays alignment width
`-ausashow`	Displays the full USA (see below) in the alignment

Feature Formats

`gff`	General Feature format defined by the Sanger Institute [default]
`embl`	Feature table used by EMBL database (em)
`swissprot`	Feature table used by SwissProt database (swiss) (sw)
`-ufo`	UFO (uniform features object) features
`-fformat`	Opens features format

These flags can be applied to the output by using "o" as a prefix, e.g. -oufo

`-fbegin`	Specifies first position
`-fend`	Specifies final position
`-freverse`	Reverses features (DNA only)

Graphic Formats

`-graph`	Static graphics using PLP plot. Output as X11 [default], PNG, ps, tektronics amongst others

Sequence Databases

Your local EMBOSS installation may have many sequence databases set up. The program `showdb` will indicate the available databases.

Uniform Sequence Address (USA)

A USA is an unambiguous means of specifying sequences in EMBOSS. It has the following syntax:

format`::`*database*`:`*entry*

Only raw (text) or IntelliGenetics format need to be specified. EMBOSS identifies the rest automatically.

You may also use:

`filename`	all sequences in a file
`filename:entry`	an entry in a file
`@list:filename`	a list file (see below)
`asis::ACGACTGACGG`	a specific short sequence

The entry can include '*' characters for wildcard matches of several entries and sequence may be specified by adding `[start:end:rev]` positions to the USA. The `rev` keyword will reverse complement a DNA sequence. Command lines using these characters must be encased in double quotes :

-D [integer] DB genetic code (def = 1).
-M [string] matrix (def = BLOSUM62).
-T [T/F] produces HTML output (def = F).
-U [T/F] uses lower case filtering (def = T) Obs.: T = any lower-case letter in input FASTA file should be masked.

Position Specific Iterated BLAST

PSI-BLAST is a variant of blast that searches a query against a database using a position-specific scoring matrix created by PSI-BLAST. First run **blastpgp** to create and save a position-specific scoring matrix, then run **blastpgp** again to search iteratively with the previously saved matrix. e.g.,

```
blastpgp -i ff.chd -d yeast -C ff.chd.ckp
blastpgp -i ff.chd -d nr -j 3 -R ff.chd.ckp
```

Selected blastpgp arguments for PSI-BLAST:

-j [integer] maximum number of iterations (def = 1).
-h [number] E-value threshold for including sequences in the score matrix model (def = 0.001).
-C [file out] stores the query and frequency count ratio matrix in a file (opt).
-Q [file out] output file for PSI-BLAST matrix in ASCII (opt).
-R [file in] restarts from a file stored previously with -C.
-B [file in] input alignment for restart.

Pattern-Hit Initiated BLAST

PHI-BLAST is a search program that combines the matching of regular expressions with local alignments surrounding the match. E.g.,:

```
blastpgp -i query.file -k pattern.file -p patseedp
```

Select blastpgp arguments for PHI-BLAST:

-i [file in] input sequence file in FASTA format.
-k [file in] pattern (syntax follows the PROSITE conventions).
-p [string] usage mode (def = blastpgp). Obs: use 'patseedp', if pattern occurs only once, and 'seedp', if it occurs more than once per protein.

Obs.: You can integrate a PSI-BLAST search after the PHI-BLAST search, using the argument "-j". E.g.,

```
blastpgp -i query -k pattern -p patseedp -j 2
```

Mega BLAST

Mega BLAST uses a greedy algorithm optimized for aligning sequences that differ slightly as a result of sequencing or other similar «errors». When a larger word size is used, it is up to 10 times faster than more common sequence similarity programs. It is also able to efficiently handle much longer DNA sequences than the blastn program.

Selected megablast arguments:

-D [integer] type of megablast output (def = 0 = alignment endpoints and score; 1 = all ungapped segments endpoints; 2 = traditional BLAST output; 3 = tab-delimited one line format).
-M [integer] maximal total length of queries for a single search (def = 20000000).
-f [T/F] shows full Ids in the output (def = F, only GIs or accessions).
-p [real] identity percentage cut off (def = 0).
-s [integer] minimal hit score to report (def = 0).

To compare two sequences

bl2seq performs a pairwise comparison between two sequences.

Selected bl2seq arguments:

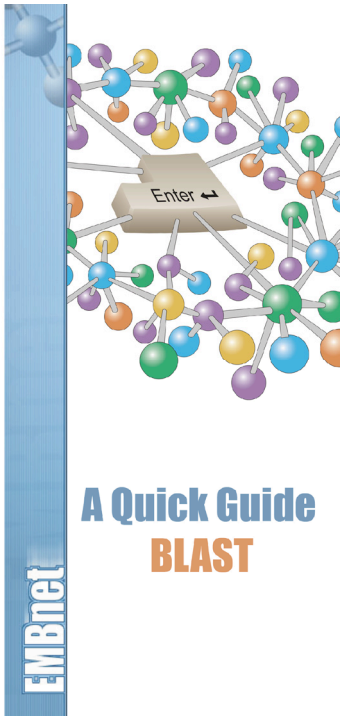
-i [file in] first sequence.
-j [file in] second sequence.
-p [string] program name (as in blastall; def = blastp).
-o [T/F] alignment output (def = stdout).
-G [integer] cost to open a gap (def = 0; zero invokes default behavior).
-E [integer] cost to extend a gap (def = 0; zero invokes default behavior).
-W [integer] wordsize (def = 0; zero invokes default behavior).
-M [string] matrix (def = BLOSUM62).
-F [string] filters query sequence (def = T).
-e [real] expectation value E (def = 10.0).
-T [T/F] produces HTML (def = F).

This document was written and designed by Eduardo Fernandes Formighieri with the help of Marcos Renato R. Araújo, Marcelo Falsarella Carazzolle and Gonçalo A. Guimarães Pereira from the Brazilian EMBnet node and distributed by the P&PR Publications Committee of EMBnet.

EMBnet – European Molecular Biology network – is a network of bioinformatics support centers situated primarily in Europe. Most countries have a national node, which can provide training courses and other forms of help for users of bioinformatics software.

http://www.embnet.org/

A Quick Guide to NCBI Blast
First edition © 2004



A Quick Guide
BLAST

A Quick Guide to the NCBI Blast

http://www.ncbi.nlm.nih.gov/blast

Blast (**B**asic **L**ocal **A**lignment **S**earch **T**ool) is a sequence comparison algorithm optimized for speed and used to search sequence databases for optimal local alignments to a query. The NCBI implementation was established by the National Center for Biotechnology Information. The program can be used through the NCBI site or can be installed locally (stand alone blast).

This guide doesn't replace the entire documentation for Blast but can be used as a reference.

Where to start?

For beginners we suggest to first read the documentation of the Blast related to similarity searching (see link below). Other useful pages are available by following the links at the top of this page.
E.g., the glossary and the tutorials:
`http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/similarity.html`

Program selection – web interface options

BLASTN – used to search nucleotide databases with a nucleotide query sequence.

MEGABLAST – a version of BLAST specially designed to efficiently find very similar sequences in a database.

Discontiguous MEGABLAST – a version of MEGABLAST used to identify similar but not identical nucleotide sequences.

Search for short nearly exact matches – used to search for primer or short nucleotide motifs in nucleotide sequences or short peptides in protein sequences.

BLASTP – used to search protein databases with a protein query sequence.

PSI-BLAST (Position-Specific Iterated BLAST) – used to search protein databases with increased sensitivity potentially locating distant homologies. A position-specific scoring matrix is created after each iteration using the selected results from the previous search.

PHI-BLAST (Pattern-Hit Iterated BLAST) – a version similar to PSI-BLAST, but including a user-defined pattern limiting the output to sequences matching the pattern. The patterns must follow the pattern syntax conventions from PROSITE.

BLASTX – makes a six-frame nucleotide query search against a protein database, finding proteins similar to those encoded by the query. Useful when the reading frame of the query is unknown or when it contains errors that may lead to frame shifts.

TBLASTN – makes a protein query search against a dynamically translated nucleotide database. Useful when searching for a specific protein against an unannotated nucleotide database, like HTGs or ESTs databases.

TBLASTX – searches al. six-frame query translations against all six-frame database translations. Effectively performs a more sensitive blastp search without doing manual translations.

CDD-Search (Conserved Domain Database Search) – used to identify conserved protein domains.

CDART (Conserved Domain Architecture Retrieval Tool) – explores the domain architectures of proteins.

Blast 2 sequences – direct comparison of two sequences.

VecScreen – screens DNA sequence queries for vector contamination using a database of known vectors.

Main databases (available at NCBI)

Protein *nr* (non-redundant + PDB + SwissProt + PIR + PRF); *swissprot* (latest major release of the SWISS-PROT); *pat* (proteins from patent division of GenBank); *month* (new data released in the last 30 days); *pdb* (3-dimensional structure records from Protein Data Bank).

Nucleotide *nr* (GenBank + EMBL + DDBJ + some PDB); *est* (GenBank + EMBL + DDBJ from EST division); *pat* (nucleotides from patent division); *pdb* (3-dimensional structure records); *month* (new data released in the last 30 days); *chromosome* (complete genomes and chromosomes); *est human* (human subset of EST); *est mouse* (mouse subset of EST); *est others* (subset of EST other than human or mouse); *gss* (Genome Survey Sequence); *htgs* (Unfinished High Throughput Genomic Sequences); *alu_repeats* (select Alu repeats from REP3ASE); *dbsts* (STS division + EMBL + DDBJ); *wgs* (assemblies of whole genome shotgun sequences).

LOCAL BLAST INSTRUCTIONS

Format source databases

formatdb formats protein or nucleotide source databases before they can be searched by blastall, blastpgp or megablast. The source database may be in either FASTA or ANS.1 format.

Selected formatdb arguments:

-t [string] title for database (opt).
-i [file in] input file for formatting.
-l [file out] logfile name (opt; def = formatdb.log).
-p [T/F] type of file (opt; T = protein (def'); F = nucleotide).
-o [T/F] parse options (opt; T = parse SeqID and create indexes; F = no parse, no indexes (def')). Obs.: the first word on the fasta definition line should be a unique identifier (SeqID).
-v [integer] size of the volume in millions of letters (opt; def = 0). Obs.: This option breaks up large FASTA files into 'volumes' (each with a maximum size of 2 billion characters). I.e.: -v 2000.
-n [string] base name for BLAST files (opt).

Fasta from databases

fastacmd retrieves FASTA formatted sequences from a BLAST database, if it was formatted using the '-o' option.

Selected fastacmd arguments:

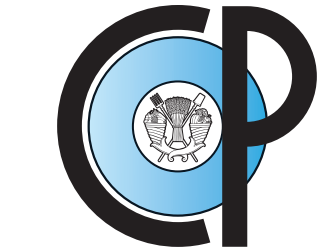
-d [string] database (def = nr).
-s [string] search string.
-i [string] input file with GIs/accessions/locuses for batch retrieval (opt).
-l [integer] line length for sequence (def = 80, opt).

Stand-alone blast

blastall performs all five flavors of blast comparison.

Selected blastall arguments:

-p [string] program name (input should be one of "blastp", "blastn", "blastx", tblastn" or "tblastx").
-d [string] database (def = nr). Obs.: Multiple database names will be accepted if quoted. E.g., -d "nr est".
-i [file in] query file (def = stdin). Obs.: Query should be in FASTA format. If multiple FASTA entries are in the input file, all queries will be searched.
-e [real] expectation value threshold (def = 10.0).
-o [file out] BLAST report output file (opt; def = stdout).
-F [string] filter query sequence (def = T). Obs.: T = DUST for blastn or SEG for others, and F = no filtering.
To change SEG options, use: `-F "S 10 1.0 1.5"`, where 10 = window value, 1.0 = low cut and 1.5 = high cut.
For coiled-coil filter: `-F "C 28 40.0 32"`, where 28 = window, 40.0 = cut off and 32 = linker.
To use both SEG and coiled-coil: `-F "C;S"`.
-b [integer] number of alignments (def = 250).
-v [integer] number of one-line description (def = 500).
-Q [integer] query genetic code (def = 1).


COLEGIO DE
POSTGRADUADOS

COLEGIO DE POSTGRADUADOS

En esta publicación intitulada "Bioinformática: aplicaciones a la genómica y proteómica", se detallan algunos de los avances más sobresalientes en los temas de genómica y proteómica, derivados de un curso internacional sobre el tema organizado por el Colegio de Postgraduados. Estos avances incluyen aspectos de las dos ciencias ómicas, incluyendo genómica y biología estructural, código R, análisis comparativo y evolución, agrupamiento y minería de datos en R, redes de interacciones entre proteínas y proteómica bioinformática.

Colegio de Postgraduados
Carretera México-Texcoco km 36.5.
Montecillo, Texcoco, Estado de México.
56230