Introducción a la Inferencia Filogenética y Evolución Molecular

23-26 Junio 2008, Fac. C. Biológicas - UANL

Pablo Vinuesa (vinuesa@ccg.unam.mx)

Centro de Ciencias Genómicas-UNAM, México
http://www.ccg.unam.mx/~vinuesa/

Todo el material del curso lo puedes descargar desde:

http://www.ccg.unam.mx/~vinuesa/UANL08

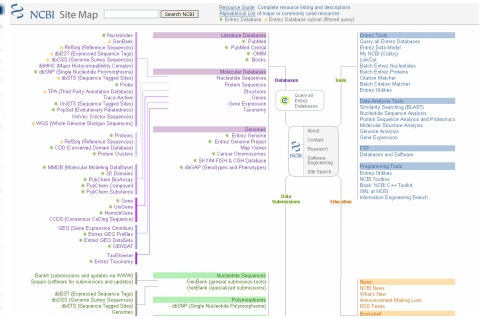
- **Tema 4: Unix/Linux y Perl en bioinformática básica – tutoriales sobre:**

1. Descarga de secuencias de GenBank usando el sistema ENTREZ
2. Uso de utilidades de UNIX para explorar los archivos fasta bajados del GB
3. Uso de Perl 1-liners para editar archivos
4. Uso de scripts de perl para automatizar procesos (ejemplos: scripts para correr Clustalw y Proml en batch

National Center for Biotechnology Information – Site map

http://www.ncbi.nlm.nih.gov/

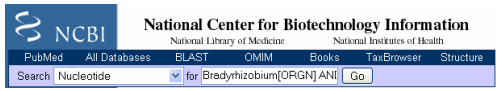
http://www.ncbi.nlm.nih.gov/Sitemap/index.html



http://www.ncbi.nlm.nih.gov/sites/gquery

Descarga de secuencias de GenBank usando el sistema Entrez

¿ Cómo recuperar todas las secuencias de *rpoB* de *Bradyrhizobium* spp. de una longitud entre 550 y 3000 nts. presentes en GenBank?



Bradyrhizobium[ORGN] AND rpoB[GENE] AND 550[SLEN]:3000[SLEN]
ó
Bradyrhizobium[orgn] AND recA[gene] NOT vinuesa[auth] AND 2007[date] NOT genome[titl]

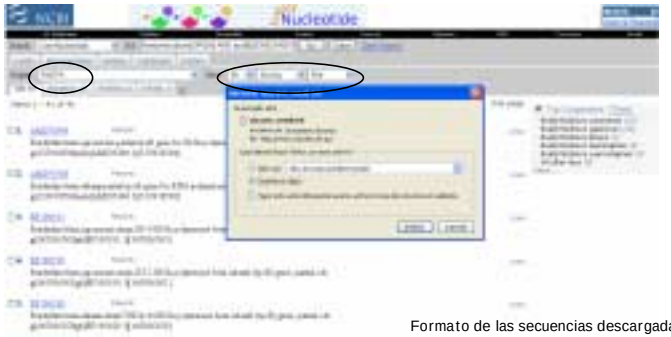
(Ver el Entrez FAQ para una lista de "keywords" usados en estas "queries" en : http://www.ncbi.nlm.nih.gov/entrez/query/static/nucprotfaq.html#B)

Descarga de secuencias de GenBank usando el sistema Entrez

No olvides de incrementar el número de entradas mostradas (por defecto sólo despliega 20) para obtener la lista completa de "hits"

Descarga de secuencias de GenBank usando el sistema Entrez

Cambia el formato a FASTA y despliega todas las secuencias, enviando el resultado a un archivo



Formato de las secuencias descargadas:

```
...
        >gi|133918594|emb|AM295349.1| Bradyrhizobium japonicum partial rpoB gene for \
         RNA polymerase beta subunit, strain LMG 6138
        TCCTATGACCAGTTCCTGATGGTCGACGAACCCGCGGGCGGGCGTCTCGACGAGGGCCTGCAGGCGGTGT
        TCCGCTCGGTGTTCCCGATCTCCGACTTCTCGGGCACCTCGATGCTGGAATTCGTCCGCTACGAATTCGA
        GCAGCCGAAATACGACGTCGACGAGTGCCGCCAGCGCGGCATGACCTTCGCGGCACCGCTCAAGGTGACG
        CTGCGCCTCATCGTGTTCGATATCGACGAGGAAACCGGCGCGAAGTCGGTGAAGGACATCAAGGAGCAGG
...
```

Formatos de secuencias
I) FASTA

- Existen una gran cantidad de estilos o formatos de presentación de secuencias. Muchos programas de análisis filogenético usan su propio formato (Phylip, Nexus, Mega ...)

- El formato más sencillo es el **FASTA**, en el que cada secuencia se identifica mediante un renglón descriptor que comienza con "**>**", y en el siguiente renglón comienza la secuencia

```
>R._galegae
CCGCTGGTCACCTCCGGCAAGCGCGCCATCCACCAGGAAGCGCCTTCCTA
CGTCGATCAGTCGACCGAAGGCCAGATCCTGGTCACCGGCATCAAGGTCG

>M._plurifarium
CCGGTCGACGCCGTCGAGCTGCGTGCCATCCACCAGCCGGCTCCGGCCTA
TGTCGACCAGTCGACGGAAGCGCAGATCCTGGTTACCGGCATCAAGGTTC

>B._japonicum
CCGGTCAAGTCGGAAGGCCTGCGCGCCATCCACCAGGAAGCGCCGACCTA
CACCGACCAGTCCACCGAAGCTGAAATTCTCGTCACCGGCATCAAGGTCG
```

Formatos de secuencias
II) PHYLIP

- **Phylip (interleaved):** no. seqs, no. caracteres
  nombre secuencias (máx 10 caracteres) espacio, secuencia ...

```
 3      100
R._galegae CCGCUGGUCA CCUCCGGCAA GCGCGCCAUC CACCAGGAAG CGCCUUCCUA
M._plurifa ...G.C.A.G ..GU..AGCU ...U..... ......CCG. .U..GG....
B._japonic ...G.CAAGU .GGAA...CU .......... .......... ....GA....

            CGUCGAUCAG UCGACCGAAG GCCAGAUCCU GGUCACCGGC AUCAAGGUCG
            U.....C... .....G.... CG........ ...U..... ........UC
            .AC...C... ..C..... CUG.A..U.. C......... .........
```

- **Phylip (sequential or non-interleaved)**

```
 3 100
R._galegae   CCGCTGGTCA CCTCCGGCAA GCGCGCCATC CACCAGGAAG CGCCTTCCTA
             CGTCGATCAG TCGACCGAAG GCCAGATCCT GGTCACCGGC ATCAAGGTCG
M._plurifa   CCGGTCGACG CCGTCGAGCT GCGTGCCATC CACCAGCCGG CTCCGGCCTA
             TGTCGACCAG TCGACGGAAG CGCAGATCCT GGTTACCGGC ATCAAGGTTC
B._japonic   CCGGTCAAGT CGGAAGGCCT GCGCGCCATC CACCAGGAAG CGCCGACCTA
             CACCGACCAG TCCACCGAAG CTGAAATTCT CGTCACCGGC ATCAAGGTCG
```

Formatos de secuencias: su interconversión

- Cuando preparamos un fichero con nuestras propias secuencias generalmente lo más adecuado es hacerlo en formato FASTA
- Si necesitamos pasarlo a otro formato, una buena opción es hacerlo con **ReadSeq**

  http://iubio.bio.indiana.edu/cgi-bin/readseq.cgi

ReadSeq reconoce automáticamente el formato de entrada y si se trata de aas o nts

- Algunos programas de inferencia filogenética como PAUP* (tiene versiones para todas las plataformas) también pueden interconvertir formatos.
  Clustal también puede hacerlo!
- También puedes usar BioPerl para hacer estas operaciones.

Uso de comandos de UNIX para inspeccionar el archivo de secuencias descargado mediante el sistema ENTREZ

- ¿ Cuantas secuencias se encuentran realmente en el archivo descargado ?

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna |wc –l # la salida de grep la filtramos con wc
  41
```

o mejor todavía:

```
$ grep -c '>' rpoB_Bradyrhizobium_550-3000.fna  # la opción –c de grep cuenta los hits de '>'
  41
```

(teclea 'man grep' y 'man wc' para abrir las correspondientes "manpages"; sin comillas)

- ¿ Cómo obtengo una lista de los GI's de los hits obtenidos ?

Primero inspeccionamos la estructura de los fastaheaders con un simple grep

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna
```

```
>gi|133918594|emb|AM295349.1| Bradyrhizobium japonicum partial rpoB gene for RNA polymerase beta subunit, strain LMG 6138
>gi|133918592|emb|AM295348.1| Bradyrhizobium elkanii partial rpoB gene for RNA polymerase beta subunit, strain LMG 6134
>gi|145926563|gb|EF190191.1| Bradyrhizobium japonicum strain X6-9 RNA polymerase beta subunit (rpoB) gene, partial cds
>gi|145926561|gb|EF190190.1| Bradyrhizobium japonicum strain X3-1 RNA polymerase beta subunit (rpoB) gene, partial cds
…
```

```
grep '>' my_downloaded_file.fasta | less  # si queremos pasar la salida de grep por un
                                          # paginador
```

Uso de comandos de UNIX para inspeccionar el archivo de secuencias descargado mediante el sistema ENTREZ

- ¿ Cómo obtengo una lista de los GI's de los hits obtenidos ? – cont.

Primero inspeccionamos la estructura de los fastaheaders con un simple grep

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna
```

```
>gi|133918594|emb|AM295349.1| Bradyrhizobium japonicum partial rpoB gene for RNA polymerase beta subunit, strain LMG 6138
>gi|133918592|emb|AM295348.1| Bradyrhizobium elkanii partial rpoB gene for RNA polymerase beta subunit, strain LMG 6134
>gi|145926563|gb|EF190191.1| Bradyrhizobium japonicum strain X6-9 RNA polymerase beta subunit (rpoB) gene, partial cds
>gi|145926561|gb|EF190190.1| Bradyrhizobium japonicum strain X3-1 RNA polymerase beta subunit (rpoB) gene, partial cds
…
```

Y vemos que el gi está después del primer pipe (|) … Por tanto podemos extractarlo fácilmente con el comando cut de la siguiente manera:

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna | cut -d'|' -f2
```

```
133918594
133918592
145926563
145926561
145926559
…
```

Y redirigir la salida a un archivo es trivial …:

```
grep '>' rpoB_Bradyrhizobium_550-3000.fna | cut -d'|' -f2  > GI_list.txt
```

Uso de comandos de UNIX para inspeccionar el archivo de secuencias descargado mediante el sistema ENTREZ

- ¿ Cómo obtengo una lista no redundante de los distintos taxa (Género y especie) presentes en el archivo FASTA descargado ?

Primero inspeccionamos la estructura de los fastaheaders con un simple grep

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna
```

```
>gi|133918594|emb|AM295349.1| Bradyrhizobium japonicum partial rpoB gene for RNA polymerase beta subunit, strain LMG 6138
>gi|133918592|emb|AM295348.1| Bradyrhizobium elkanii partial rpoB gene for RNA polymerase beta subunit, strain LMG 6134
>gi|145926563|gb|EF190191.1| Bradyrhizobium japonicum strain X6-9 RNA polymerase beta subunit (rpoB) gene, partial cds
>gi|145926561|gb|EF190190.1| Bradyrhizobium japonicum strain X3-1 RNA polymerase beta subunit (rpoB) gene, partial cds
…
```

Y vemos que el nombre del género (*Bradyrhizobium*) y del epíteto específico (*japonicum*) para este primer fasta header (y las demás líneas) vienen justo después del 1er. y 2do. campos blancos (espacios sencillos). Por tanto usamos **cut** con el delimitador puesto a Espacio en blanco (**-d' '**) quedándonos con el 2º y 3er campo, respectivamente **-f2,3**. Luego los ordenamos con **sort** y nos quedamos con los nombres únicos (**uniq**)

```
grep '>' rpoB_Bradyrhizobium_550-3000.fna | cut -d' ' -f2,3 | sort | uniq
```

Y contarlos es trivial, sólo le añadimos un wc (wordcount –l que cuenta las líneas) …:

```
grep '>' rpoB_Bradyrhizobium_550-3000.fna | cut -d' ' -f2,3 | sort | uniq | wc -l
```

Uso de comandos de UNIX para inspeccionar el archivo de secuencias descargado mediante el sistema ENTREZ

• ¿ Cómo obtengo una lista no redundante de los distintos taxa (Género y especie) presentes en el archivo FASTA descargado ?

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna | cut -d' ' -f2,3 | sort | uniq
Bradyrhizobium canariense
Bradyrhizobium elkanii
Bradyrhizobium genosp.
Bradyrhizobium japonicum
Bradyrhizobium liaoningense
Bradyrhizobium sp.
Bradyrhizobium yuanmingense
```

Y contarlos es trivial, sólo le añadimos un wc (wordcount –l que cuenta las líneas) ...:

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna | cut -d' ' -f2,3 | sort | uniq | wc –l
7
```

Usa Perl para facilitar el trabajo rutinario

Para una introducción muy corta al poderoso lenguaje de programación (scripting) **Perl** (**Practical Extraction and Report Language** or **Pathologically Ecclectic Rubish Lister**) pueden consultar los siguientes tutoriales disponibles desde **perl.org** y **perl.com**:

http://perldoc.perl.org/perlintro.html

y

http://www.perl.com/pub/a/2000/10/begperl1.html
http://www.perl.com/pub/a/2000/11/begperl2.html
http://www.perl.com/pub/a/2000/11/begperl3.html

• Estos tutoriales los puedes descargar también desde el sito web de nuestro curso

En **perl.org** y **perl.com** pueden encontrar tutoriales más avanzados sobre tópicos específicos de gran importancia como expresiones regulares, subrutinas, referencias, estructursa de datos, paquetes, módulos, objetos, programación de bases de datos a través de DBI, Perl e internet ...
Una vez instalado Perl, tienes toda la documentación, incluyendo FAQs instalados. Explóralos usando los comandos **perldoc perl; ó perldoc perlintro ... etc.** Para consultar la ayuda sobre una función específica teclea por ejemplo **perldoc –f split**

Usa Perl para facilitar el trabajo rutinario

Escribe tus scripts de Perl usando tu editor favorito com vi, emacs o nedit.

Los scripts de Perl suelen nombrarse con la terminación **.pl** (miscript.pl) ó **.plx**

Para ejecutar miscript.pl puedes hacerlo de varias maneras:

desde el directorio en el que está el script teclea por ejemplo perl miscript.pl ARG1 ... > output

Generalmente los scripts de Perl (o de otros lenguajes de scripting) llevan como primera línea la "shebang line" que apunta al directorio del intérprete perl:

```
#!/usr/bin/perl –w  # (usa 'which perl' para saber la ruta a perl de tu máquina)
```

Para hacer miscript.pl ejecutable tienes que usar el comando **chmod +x miscript.pl** (si prefieres usa **chmod 755 miscript.pl**). Si haces ahora un **ls –l** verás que miscript.pl ya es ejecutable. Entonces lo puedes correr así: **./miscript.pl**

Si pones miscript.pl en un directorio que esté en el **search path** (ruta de búsqueda) entonces puedes invocar el script directamente como miscript.pl desde la línea de comandos, es decir, como cualquier otro comando Unix. (teclea **echo $PATH** en tu máquina)

Usa Perl para facilitar el trabajo rutinario: Perl one-liners

• Para muchos trabajos puntuales como edición y parseo de archivos de texto es muy práctico saber usar Perl desde la línea de comandos.
ver http://www.perl.com/lpt/a/857 (Perl command-line options; un sencillo tutorial)

```
$ perl -e 'print "hola tronco!\n"'     # perl –e 'el código va entre comillas sencillas'
$ perl -e 'print "hola "' -e 'print "tronco!\n"'  # puedes usar múltiples -e
```

• **-n switch**

```
$ perl -n -e 'tu código' archivo1  # ojo, -e siempre junto a la línea de código
Perl interpreta esta línea como:
LINE:
while (<>) { # lee desde STDIN y se porcesa el input línea por línea
   # tu código
 }
```

• **-p switch**

```
$ perl -p -e 'tu código' archivo1
LINE:
while (<>) {
   # tu código va aquí
} continue {
   print or die "-p destination: $!\n";
}
```

Usa Perl para facilitar el trabajo rutinario: Perl one-liners

- Usa perl –pe para editar a tus archivos

```
$ perl -pe 'tu código' < input.txt > output.txt
```

- Usa **perl –i.ext –pe 'tu código' archivo.txt** para obtener un archivo editado y una copia del original con la extensión dada a –i (OJO, NO OLVIDES PONER LA EXTENSION O NO SE GENERA EL BACKUP!!!!)

- **Veamos unos one-liners en acción.** Vamos a modificar el "fastaheader" del archivo rpoB_Bradyrhizobium_550-3000.fna que descargamos mediante el sistema ENTREZ. Recuerda que puedes ver su estructura con el siguiente comando:

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna
```

```
>gi|133918594|emb|AM295349.1| Bradyrhizobium japonicum partial rpoB gene for RNA polymerase beta subunit, strain LMG 6138
>gi|133918592|emb|AM295348.1| Bradyrhizobium elkanii partial rpoB gene for RNA polymerase beta subunit, strain LMG 6134
>gi|145926563|gb|EF190191.1| Bradyrhizobium japonicum strain X6-9 RNA polymerase beta subunit (rpoB) gene, partial cds
>gi|145926561|gb|EF190190.1| Bradyrhizobium japonicum strain X3-1 RNA polymerase beta subunit (rpoB) gene, partial cds
...
```

Queremos que las líneas queden de la siguente manera:

```
>NO_GI [ Bradyrhizobium japonicum LMG6138 ]
```

Usa Perl para facilitar el trabajo rutinario: Perl one-liners

```
$ grep '>' rpoB_Bradyrhizobium_550-3000.fna
```

```
>gi|133918594|emb|AM295349.1| Bradyrhizobium japonicum partial rpoB gene for RNA polymerase beta subunit, strain LMG 6138
>gi|133918592|emb|AM295348.1| Bradyrhizobium elkanii partial rpoB gene for RNA polymerase beta subunit, strain LMG 6134
>gi|145926563|gb|EF190191.1| Bradyrhizobium japonicum strain X6-9 RNA polymerase beta subunit (rpoB) gene, partial cds
>gi|145926561|gb|EF190190.1| Bradyrhizobium japonicum strain X3-1 RNA polymerase beta subunit (rpoB) gene, partial cds
...
```

Queremos que las líneas queden de la siguente manera:

```
>NO_GI [ Bradyrhizobium japonicum LMG6138 ]
```

- Una solución es la siguiente (escríbelo todo en una sola línea o usa \ para saltos de línea):

```
cat rpoB_Bradyrhizobium_550-3000.fna | perl -p -e 's/^>gi\|/>/; s/\|\w+\|.*\|//;
s/RNA|RpoB.*//; s/^>(\d+)\s+(.*)/>$1 \[$2\]/g;
s/partial rpoB gene for  polymerase beta subunit,//g;
s/polymerase beta subunit \(rpoB\) gene, partial cds//g; s/\s+\]/\]/g; s/strain //' | grep '>
```

```
>133918594 [Bradyrhizobium japonicum  LMG 6138]
>133918592 [Bradyrhizobium elkanii  LMG 6134]
>145926563 [Bradyrhizobium japonicum X6-9]
>145926561 [Bradyrhizobium japonicum X3-1]
>145926559 [Bradyrhizobium elkanii USDA 94]
>145926557 [Bradyrhizobium elkanii USDA 76]
>145926555 [Bradyrhizobium elkanii USDA 46]
>145926553 [Bradyrhizobium japonicum USDA 122]
>145926551 [Bradyrhizobium yuanmingense TAL760]
...
```

Usa Perl para facilitar el trabajo rutinario: Perl one-liners

- Hay que ir probando pedazo a pedazo el resultado del código. Filtra la salida con grep '>' para ver el efecto sobre las líneas del "fastaheader" de cada sentencia (separadas por ;). Una vez que estés satisfecho con el resultado lo rediriges a un archivo (> ed_output_file)

```
perl -p -e 's/^>gi\|/>/; s/\|\w+\|.*\|//;
s/RNA|RpoB.*//; s/^>(\d+)\s+(.*)/>$1 \[$2\]/g;
s/partial rpoB gene for  polymerase beta subunit,//g;
s/polymerase beta subunit \(rpoB\) gene, partial cds//g; s/\s+\]/\]/g; s/strain //'
rpoB_Bradyrhizobium_550-3000.fna  > rpoB_Bradyrhizobium_550-3000_ed.fna
```

Fíjate que es equivalente escribir:

```
cat mi_archivo.fas | perl –pe 'mi_código' | grep '>'
                    ó
perl –pe 'mi_código'  mi_archivo.fas | grep '>'
```

Con un poco de práctica verás lo útiles que son estos 1-liners ...

Libros de referencia recomendados

Uno de los aspectos más importantes de Perl es la impresionante cantidad de código disponible en forma de módulos en **CPAN** y específicamente para bioinformática en el sitio del proyecto **BioPerl** (busca estos términos en google)

Practical Extraction and Report Language
(" Pathologically Ecclectic Rubish Lister")

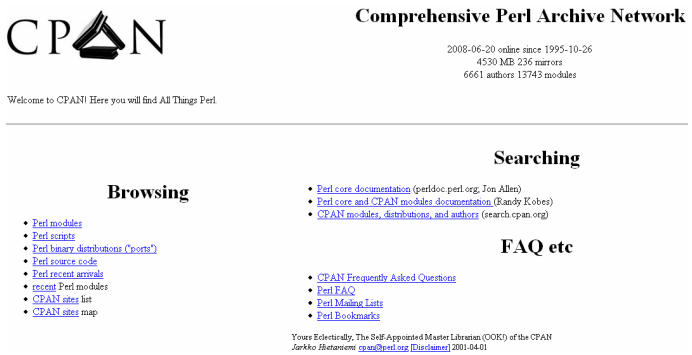
- **Beginning Perl programming:**

O´Reily:
Learning Perl 4th ed.
Beginning Perl for Bioinformatics

- **Advanced Perl programming:**

O´Reily:
Programming Perl 3rd ed.
Perl Cookbook 2cnd ed.
Perl Best Practices
Mastering Perl for Bioinformatics

http://www.cpan.org/



http://www.cpan.org/

http://perldoc.perl.org/perlintro.html

Estudia este minitutorial antes de ver el código que se presentará más adelante!!!

http://perldoc.perl.org/index-tutorials.html

http://search.cpan.org/

http://www.bioperl.org/wiki/Main_Page


```
#!/usr/bin/perl -w

#-------------------------------------------------------------------------------------------------------#
#  run_clustalw_V01.pl written by P. Vinuesa 26-04-2006
#  vinuesa@ccg.unam.mx; http://www.ccg.unam.mx/~vinuesa
#  Centro de Ciencias Genomicas-UNAM, Mexico
#-------------------------------------------------------------------------------------------------------#

# This is a heavily commented script intended to teach you some basic Perl functions and idioms

# The script will align all files of a particular type specified as an argument at the command line
# (like fas or txt ...), present in the working or current directory, using default parameters.
# The output will be converted to the selected output format, provided as a second argument at the
# command line.
# After performing these multiple alignments, the program will ask you whether you want to perform
# a profile alignment of a pair of selected aligned files, again providing the option of specifying a particular
# output format.

# The script assumes that clustalw 1.83 or higher is found in the search path!

# run from within the directory containing the sequence files to align

# usage: perl run_clustalw_V01.pl <[fas|fasta]> and  <[clustal|fasta|phylip|nexus]>
```

```
#-------------------------------------------------------------------------------------------------------#

my $version = '0.0.3'; # March 08, minor revision of comment texts;
                       # progress reports and directory cleanup added.

use strict;  # always be strict. Well, in short scripts as this one it may be a bit overkill... but use it anyway!
use File::Basename; # calls a standard Perl module

my $progname = basename($0); # $0 is a SHELL variable that holds the program name; we get rid of the
                             # path to $0 by processing it with the basename() subroutine of the
                             # File::Basename module.


#------------------------------------ MAIN PROGRAM CODE-------------------------------------------#

# 0) die program and print an error message if the proper arguments are not provided at the cmmd line

die "\n\tusage: $progname V:$version needs two args: [fas|fasta|txt] and [clustal|fasta|phylip|nexus],
        which define the extension of the input files to align
            and the desired output format for the multiple sequence alignments\n\n\n" unless $ARGV[1];

# 1) declare lexical (my) variables;

my(@files, $file, @parts, $basename, $file_ext, $short_ext, $output_format,
    $processed_alignments_counter, $cwd, @trash, $no_of_seqs,);

my($answer, $profile1, $profile2, $basename1, $ext1, $basename2, $ext2,);
```

```
# 2) copy the (positional) command line arguments saved in the special array @ARGV into named variables


$cwd = `pwd`;   # backticks read the output of shell commands, assigining the result to to a scalar variable
chomp $cwd;     # chomp() removes trailing "\n" (newline characters).
($file_ext, $output_format) = @ARGV;
$short_ext = substr($output_format, 0, 3); # the substr() function will extract chuncks of text from a string:
                                           # substr(TARGET, OFFSET, LENGTH, REPLACEMENT)
                                           # in our example only TARGET, OFFSET and LENGTH are specified.
                                           # If $output_format contains fasta -> $short_ext will contain fas.
```

```
# 3) Iterate over the specified input files. They enter one by one into the while loop carried in the $file scalar
#    and are then passed to clustalw, which is called with Perl's system function. Perl passes the job
#    to the shell and sleeps until the clustalw process has finished.

while(defined($file=glob("*.$file_ext"))){ # here we use the glob() function to "expand" all filenames with the
                                           # supplied file_ext in the cwd, passing them one by one to $file
   $no_of_seqs = `grep -c '>' $file`;      # determine the no. of seqs. in file $file using grep -c enclosed in backticks
   chomp $no_of_seqs;                      # remove the trailing new line character.
   @parts=split(/\./,$file);               # here we split the file names at the dot character. (e.g. myfile.fas gets split into
                                           # myfile and fas)
   $basename=$parts[0];                    # the first element of the @parts array contains the file's "basename" (i.e. myfile)
   # $basename= (split(/\./,$file))[0];    # This is a more perlish and succint way of writing the above two lines
                                           #  ... or even ($basename, $ext) = (split(/\./,$file))[0,1];
   if($output_format ne 'clustal'){
            print "\n\t# $progname is aligning $no_of_seqs sequences from the input file $file ...";
      system ("clustalw -infile=$basename\.$parts[1] -align -outorder=aligned -outfile=$basename\_aln1.$short_ext
               -convert -output=$output_format >&n"); # ojo, toda la sentencia debe de ir escrita en una sola línea!
            $processed_alignments_counter++;
   }
   else{
      print "\n\t# $progname is aligning $no_of_seqs sequences from the input file $file ...";
            system ("clustalw -infile=$basename\.$parts[1] -align -outorder=aligned -outfile=$basename\_aln1.aln >&n");
            $processed_alignments_counter++;
   }
}


 print "\n\n\t# $progname finished generating $processed_alignments_counter multiple sequence alignments
                                        available in $output_format format\n\t\tthe results are found in DIR $cwd\n\n";
```

```
# 4) and now we can add an option to do pairwise profile alignments. We use the STDIN filehandle to get
#    the user input to the script directly from the cmmd line. We chomp the cmmd line args to get rid of
#    the newline ("\n") character introduced after hitting the return button. We also use a regular expression ~/y/i
#    (matches Y or y; the 'i' modifier stands for ignore case) and the match operator =~ (read the if ($answer =~ /y/i){} line
#    as: if the answer provided by the user matches y or Y, then execute the block {}, otherwise (else) exit the program.
#    The system() function returns control of the program to the shell. Just type regular shell commands within double quotes
#    you want the program to execute. Perl sleeps until the shell has finished the jobs.

print "\t\t~ Do you want to perform a profile alignment? type: Y or N\n\n";
chomp($answer= <STDIN>);
if ($answer =~ /y/i){
   print "\t\t\t~ provide the name of the first profile alignment\n";
   chomp($profile1=<STDIN>);

   print "\t\t\t~ provide the name of the second profile alignment\n";
   chomp($profile2=<STDIN>);

   print "\t\t\t~ provide the output format for the multiple aln [clustal|fasta|phylip|nexus]\n";
   chomp($output_format=<STDIN>);

   $short_ext = substr($output_format, 0, 3);
   ($basename1, $ext1)=split(/\./,$profile1);
   ($basename2, $ext2)=split(/\./,$profile2);
   if($output_format ne 'clustal'){
      print "\n\t# $progname is making the profile alignment of files $profile1 and $profile2 ...";
            system ("clustalw -profile1=$basename1\.$ext1 -profile2=$basename2\.$ext2 -profile
                              -outfile=combined_profiles.$short_ext -convert -output=$output_format >&n");
   }
   else{
      print "\n\t# $progname is making the profile alignment of files $profile1 and $profile2 ...";
            system ("clustalw -profile1=$basename1\.$ext1 -profile2=$basename2\.$ext2 -profile
                                                          -outfile=combined_profiles.aln >&n");
   }
   print "\n\t# $progname finished the profile alignment of files $profile1 and $profile2 available
                                                 in $output_format format\n\t\tthe results are found in DIR $cwd\n\n";
}
```

```
# ... viene de la página anterior


# 1) Declare variable, get input arguments from the command line and print help if needed ... continuation

my($inputformat, $infile_ext, $outputformat, $outputfile_ext)=@ARGV;
my @infiles = <*$infile_ext>;
my($basename,$counter);

# 2) Process all files in cwd having infile_ext, converting them from inputformat to outputformat
#    with the outputfile_ext provided at the command line

foreach my $infile ( @infiles )
{
      $basename = (split(/\./, $infile))[0];

      my $in  = Bio::AlignIO->new(-file => $basename . ".$infile_ext",   -format => $inputformat);
      my $out = Bio::AlignIO->new(-file => ">$basename.$outputfile_ext", -format => $outputformat);

      while ( my $aln = $in->next_aln() )
      {
            $out->write_aln($aln);
      }
      $counter++;

}

# 3) Print a final summary report
print "\n\t# $0 is done: $counter $inputformat input files with $infile_ext extension were converted
to $outputformat format with $outputfile_ext extension\n\n";
```

```
# 5) and finally make a directory cleanup, reporting which files were removed

# opendir() opens a directory filehandle 'DIR' to read from
# readdir() allows to read from the DIR filehandle, filtering with grep the files n and *.dnd, which are saved
# in @trash. Unlink is the perlish way to remove (delete) files.
# (be careful, the delete builtin function deletes a key and its value  from  a %hash,
# but not files from a directory!


opendir(DIR, $cwd);

@trash = grep { /\.dnd$|^n$/ } readdir(DIR);

if( @trash ) {
   print "\n\t# $progname has removed ", scalar(@trash), " temporary files:\n\t\t", join("\n\t\t", @trash),
                                                                                    "\n\n\n\t\t ... and is exiting now!\n\n";
   unlink @trash
}
```

Y ahora un ejemplo de cómo usar el módulo Bio::AlignIO de BioPerl para hacer la interconversión entre formatos para múltiples archivos de secuencia en batch:

```
#!/usr/bin/perl -w
use strict;
use Bio::AlignIO;
# convert_aln_format_batch_bp.pl, por P. Vinuesa, CCG-UNAM; http://www.ccg.unam.mx/~vinuesa
# requires that the Bio::AlignIO module from the BioPerl core distribution is installed on your system

# 1) Declare variable, get input arguments from the command line and print help if needed

if ($#ARGV < 3)
{
      print "\nUsage: $0 inputformat infile_ext outputformat outputfile_ext\n
            Supported formats include:

         bl2seq      Bl2seq Blast output
         clustalw    clustalw (.aln) format
         emboss      EMBOSS water and needle format
         fasta       FASTA format
         maf         Multiple Alignment Format
         mase        mase (seaview) format
         mega        MEGA format
         meme        MEME format
         msf         msf (GCG) format
         nexus       Swofford et al NEXUS format
         pfam        Pfam sequence alignment format
         phylip      Felsenstein PHYLIP format
         prodom      prodom (protein domain) format
         psi         PSI?BLAST format
         selex       selex (hmmer) format
         stockholm   stockholm format\n\n\n";
      exit;
}
```