

Anexos con programas y material complementario:

+ANEXO I (Programas y módulos ya existentes):

-CD-HIT-EST:

1

- cd-hit-est -i fasta.fa -o outfasta -l 200 -c 0.97 -M 4000 -aL 0.8 -aS 0.8 -r
- -i :archivo de entrada
- -o :archivo de salida
- -l :longitud mínima de secuencias
- -c :threshold
- -M :uso de memoria máxima
- -aL :coverage de la cadena larga
- -aS :coverage de la cadena corta
- -r para comprobar las cadenas en +/+ y +/-

-BLAST:

-a 2

- blastall -p blastn -i outfasta -d greengenes -o file.out -m 8 -e 1e-03 -D 200
- -p : el programa blastn para nucleotidos
- -i : el archivo de entrada (un fasta)
- -d :la base de datos que se va a utilizar para lanzar las secuencias
- -o : el archivo de salida.
- -m : formato del archivo de salida
- -e : el evalúe mínimo
- -D : la cantidad de salidas para cada entrada
- -a : la cantidad de cpu a utilizar.

-Text-LevenshteinXS:

-<http://search.cpan.org/dist/Text-LevenshteinXS/LevenshteinXS.pm>

- Para instalar un módulo cpan:
- \$ sudo perl -MCPAN -e shell
- entrar en la shell, e instalar el módulo Text-LevenshteinXS :
- \$ install Text-LevenshteinXS

-asigna16S.pl:

```
#!/usr/bin/perl
```

```
-- Asigna especies a secuencias 16S.
```

```
-- Javier Tamames Jan 2010
```

```
$|=1;
```

```
use Tie::IxHash;
```

```
$homedir="/home/master/Mahaigaina/blast";      -- Modificar segun  
donde se este utilizando
```

```
$minevalue=1e-03;
```

```
$minidentity=90;
```

```
$excludenotax=0; -- A uno, considera los hits sin taxon para la clasificacion  
(puede dar muchos "none")
```

```
-- A cero, los ignora (puede empezar a considerar hits que estan  
muy abajo en el Blast)
```

```
$method="bestaver"; #-- "besthit" o "bestaver"
```

```
$database="greengenes"; #-- silva o greengenes
```

```
#-- Parametros para bestaver
```

```
$mindiff=0.02; #-- Diferencia de puntuacion para asignarlo al mejor
```

```
$numtaxhits=5; #-- Los primeros hits que se toman para calcular la  
puntuacion media de cada taxon
```

```
$mintaxhits=3; #-- Numero minimo de hits con un taxon para considerarlo
```

```
@ranks=("superkingdom","phylum","class","order","family","genus","species")  
;
```

```
$blastfile=$ARGV[0];
```

```
if(!$blastfile) { die "Usage: perl asigna16S_GG.pl <blast file>\n"; }
```

```
#$fastafile="/home/tamames/borrar/A.fasta";
```

```
#$blastfile="/home/tamames/borrar/A.out";
```

```
readtax($database); #-- Leer taxones de greengenes o silva
```

```
main();
```

```
output(); #-- Si no se quiere ficheros de salida, comentar esta linea
```

```
sub main {
```

```
open(in,$blastfile) || die;
```

```
print "# Creado por $0, ",scalar localtime()."; Blast file: $blastfile; Database:  
$database\n";
```

```
print "# Method: $method";
```

```
if($method ne "besthit") { print "; mindiff=$mindiff;  
numtaxhits=$numtaxhits; mintaxhits=$mintaxhits"; }
```

```
print "\n";
```

```
while(<in>) {
```

```
chomp;
```

```
next if !$_;
```

```
@fields=split(/\t/,$_);
```

```
$current=$fields[0];
```

```
if($last && ($current ne $last)) {
```

```
printresults();
```

```
(%acpositions,%total,%aver,%bbest,%miden,%idmax)=();
```

```
($maxbit,$besthit,$bestcoghit)="";
```

```
$position=0;
```

```
}
```

```

$last=$current;

if(!$maxbit) { $maxbit=$fields[11]; }

next if !$maxbit;

$evalue=$fields[10];

$bitscore=$fields[11];

$normscore=$bitscore/$maxbit;

$identity=$fields[2];

$gname=$tax{$fields[1]};

$queryname=$truetax{$current};

# next if($identity==100);      #-- Excluye identicos

next if($evalue>$minevalue);

next if($identity<$minidentity);

# print "**$current $fields[1] $gname $normscore $identity\n";

$position++;

if(!$besttaxhit && $gname) { $besttaxhit=$gname; $maxident=$identity; }

$acpositions{$normscore}{$gname}++;

@ttx=split(/\./,$gname);

foreach my $nm(@ttx) {

    $txname=$nm;

```

```
$txname=~s/\(.*//;
```

```
        if(!$miden{$txname} || ($miden{$txname}<$identity)) {  
$miden{$txname}=$identity; }  
  
        }  
  
    }  
  
}
```

```
close inc;
```

```
printresults();
```

```
($last,$current)="";
```

```
(%acpositions,%total,%aver,%bbest,%miden,%idmax)=();
```

```
($maxbit,$besthit,$besttaxhit)="";
```

```
$position=0;
```

```
}
```

```
sub printresults {
```

```
    print "\n";
```

```
    for my $thisrank(@ranks) {
```

```
        ($truet,$tbesthit)="";
```

```
        @ttx=split(/\;/,$truetax{$last});
```

```

for my $lrank(@ttx) {

    if($lrank=~/\($thisrank\)/) { $truet=$lrank; }

    }

# next if(!$truet);

if(!$besttaxhit) { print "$last\t$thisrank\t$truet\tNo hit\n"; return; }

(%aver,%total)=();

($bestsc,$secondsc,$besth,$secondh)="";

foreach my $ascores(sort { $b<=>$a; } keys %acpositions) {

    foreach my $ahits(sort keys %{ $acpositions{$ascores} }) {

        @tfid=split(/\;/,$ahits);

        # $tuiden=$miden{$ahits};

        $tbesthit="";

        for my $lrank(@tfid) {

            if($lrank=~/\($thisrank\)/) {

                $lrank=~s/\($thisrank\)//;

                if(!$tbesthit) {

                    $tbesthit=$lrank;

                    $tuiden=$miden{$tbesthit};

                }

```

```
}
```

```
}
```

```
if(!$tbesthit) {
```

```
    if($excludenotax) { $tbesthit="None"; }
```

```
    else { next; }
```

```
}
```

```
    # print "$last $thisrank $tbesthit $lrank $ahits $ascores =>  
$acpositions{$ascores}{$ahits} $idmax{$thisrank}\n";
```

```
for(my $cou=1; $cou<=$acpositions{$ascores}{$ahits}; $cou++) {
```

```
    if($total{$tbesthit}<$numtaxhits) {
```

```
        $aver{$tbesthit}+=$ascores;
```

```
        $total{$tbesthit}++;
```

```
        # print "*** $tbesthit $ascores $aver{$tbesthit} $total{$tbesthit}  
$tuiden\n";
```

```
        if(!$idmax{$tbesthit}) { $idmax{$tbesthit}=$tuiden; }
```

```
    }
```

```
}
```

```
}
```

```
}
```

```
foreach my $nm(sort keys %aver) { $aver{$nm}/=$total{$nm}; }
```

```
foreach my $print(sort { $aver{$b}<=>$aver{$a}; } keys %aver) {
```



```

# printf "$last\t$print\t$total{$print}\t%.3f\n",$aver{$print};

if(!$besth) { $besth=$print; $bestsc=$aver{$print}; }

elseif(!$secondsc) { $secondsc=$aver{$print}; $secondh=$print; }

}

if(!$secondsc) { $secondsc=0; }

$difscore=$bestsc-$secondsc;

$bprint=$besth;

$bprint=~s/\(.*//g;

if(($difscore>=$mindiff) && ($total{$besth}>=$mintaxhits)) { print
"$last\t$thisrank\t$bprint\t$bprint\t$idmax{$besth}\n"; $rta=$besth; }

else {

print "$last\t$thisrank\tUnresolved\t$bprint\t$idmax{$besth}\n";

if($method eq "bestaver") { $rta="Unresolved"; } else { $rta="$besth"; }

}

$sum{$thisrank}{$rta}++;

# else { print "$last\t$thisrank\t$truet\tUnresolved $bestsc $besth $secondsc
$secondh\n"; }

}

}

sub readtax {

```

```

$datab=shift;

if($datab eq "greengenes") { $tufile="$homedir/greengenes.tax"; }

elsif($datab eq "silva") { $tufile="$homedir/silva.tax"; }

open(in,$tufile) || die;

while(<in>) {

chomp;

next if(!$_ || ($_ =~ /^\/\#/));

if($_ =~ /^\/\>/) {

@afi=split(/\/t/, $_);

@ids=split(/\/s+/, $afi[0]);

$id=$ids[0];

$id =~ s/^\/\>//;

$curtax=$afi[1];

$tax{$id}=$curtax;

$truetax{$id}=$curtax;

    }

}

close in;

}

```

```

sub output {

    foreach my $tax(sort keys %sum) {

        $outfil="$homedir/$tax.xls";

        open(outr,">$outfil") || die;

        foreach my $ct(sort keys %{ $sum{$tax} }) { print outr "$ct\t$sum{$tax}
{$ct}\n"; }

        close outr;

    }

}

```

-mtax.pl:

```
#!/usr/bin/perl
```

```
# Crea lista de taxones para cada id de una base de datos (greengenes)
```

```
# Crea fichero necesario para el uso con asigna16.pl
```

```
$|=1;
```

```
use taxbuild_NOT_EUK;
```

```

my $taxDB = taxbuild_NOT_EUK->new(

    nodes => "/home/tamames/databases/taxonomy/nodes.dmp",    # archivo
de la base de datos taxonomy del ncbi

    names => "/home/tamames/databases/taxonomy/names.dmp"    # archivo
de la base de datos taxonomy del ncbi

);

$data="/home/tamames/databases/current_prokMSA_unaligned.fasta";
# archivo de la base de datos del greengenes

$data="/home/tamames/temp/current_prokMSA_unaligned.fasta";


print "# Created by $0, ",scalar localtime," from $data\n";

open(in,$data) || die;

while(<in>) {

    chomp;

    next if !$_;

    if($_=~/^>/) {

        ($sid,$gbid,$spec)=split(/\s+/, $_,3);

        @nf=split(/\s+/, $spec);

        $sname="$nf[0] $nf[1]";

        $tax=new($sname);

```

```

if($tax=~/\w/) { print "$sid\t$tax\t$spec\n"; }

    }

}

```

```

sub new {

```

```

    my $rec=shift;

```

```

    my @keynames = split /\s+/, $rec;

```

```

    my $keyTaxID;

```

```

    # print ">>>>>>>>>$rrr\n";

```

```

    if(!$rec) { return; }

```

```

    if($taxseen{$rec}) { return $taxseen{$rec}; }

```

```

    # print "Probing $rec\n";

```

```

    eval {$keyTaxID = $taxDB->get_taxid_from_name($rec)};

```

```

        if(!$keyTaxID) { eval {$keyTaxID = $taxDB->get_taxid_from_name($keynames[0])}; }

```

```

    my @taxonomy = $taxDB->get_taxonomy_with_levels($keyTaxID);

```

```

    $fin=join ";;",map {"$_->[0]($_->[1])"} @taxonomy;

```

```

    $taxseen{$rec}=$fin;

```

```

    return $fin;

```

```

    }

```

-taxbuild_NOT_EUK.pm:

```
#!/usr/bin/perl
```

```
package taxbuild_NOT_EUK;
```

```
=head1 NAME
```

```
taxbuild - Get the taxonomy of a gene
```

```
=head1 SYNOPSIS
```

```
use taxbuild;
```

```
my $taxDB = taxbuild->new(
```

```
    nodes => $nodesFile,
```

```
    names => $namesFile,
```

```
    dict => $dictFile,
```

```
    save_mem => 0
```

```
);
```

```
# Get the taxonomy given a GI identifier
```

```
my @tax = $taxDB->get_taxonomy_from_gi("35961124");
```

```
# Get the taxonomy term of a GI identifier at a given level
```

```
my $term_at_level = $taxDB->get_term_at_level_from_gi("35961124","family");
```

```
# Get the taxid of a GI identifier
```

```
my $taxid = $taxDB->get_taxid("35961124");
```

```
# Get the taxonomy given a taxid
```

```
my @tax = $taxDB->get_taxonomy($taxid);
```

```
# Get the taxonomy at a given level given a taxid
```

```
my $taxid_at_level = $taxDB->get_term_at_level($taxid,"genus");
```

```
# Get the level of a given taxonomical name
```

```
my $level = $taxDB->get_level_from_name("Proteobacteria");
```

=head1 DESCRIPTION

This module has been designed to easily retrieve the taxonomy of a gene fast and with low memory usage.

It parses the taxonomy db that can be downloaded from NCBI at the following address: [L<ftp://ftp.ncbi.nih.gov/pub/taxonomy/>](ftp://ftp.ncbi.nih.gov/pub/taxonomy/)

You need to pass to the constructor (the "new" method) the following:

=over 4

=item * The I<nodes> file of the taxonomy database. Both the name of the file or its filehandle if the file is already open are allowed. This file is located at I<taxdump.tar.gz>

=item * The I<names> file of the taxonomy database. Both the name of the file or its filehandle if the file is already open are allowed. This file is located at I<taxdump.tar.gz>

=item * (optional) The I<dict> file containing the correspondences between B<gi> and B<taxIds>. This file can be downloaded from the taxonomy database too (it is called I<gi_taxid_prot.dmp.gz>), but you can't use it directly with this module, instead you need to convert it to binary format. This conversion improves dramatically the speed of the script and saves a lot of memory. There should be an accompanion script with this module called I<tax2bin.pl> that can do this task.

=item * (optional) I<save_mem>. If you want to save some memory, the correspondencies between B<gi> and B<taxIds> will not be loaded into memory. This will save aprox 800Mb of system memory, but looking up for a taxId will be ~20% slower. Note that this option is I<off> by default and only has sense if you are using the I<dict> option too.

=back

=head1 METHODS

Currently this module provides the following methods:

=head2 C<get_taxid>

Given a I<gi> number, returns its taxId

```
my $taxid = $taxDB->get_taxid($gi);
```

=head2 C<get_taxonomy_from_gi>

Given a I<gi> number, returns an array with its taxonomy

```
my @taxonomy = $taxDB->get_taxonmy_from_gi($gi);
```

```
my $taxonomy = join ":",@taxonomy;
```

```
=head2 C<get_taxonomy>
```

Given a taxId number, returns an array with its taxonomy

```
my @taxonomy = $taxDB->get_taxonomy($taxid);
```

```
=head2 C<get_term_at_level_from_gi>
```

Returns the taxonomy term of the gene at a given level

```
my $family = $taxDB->get_term_at_level($gi,"family");
```

```
$families{$family++};
```

```
=head2 C<get_term_at_level>
```

Returns the parent term of a given taxonomy given the taxId and the level

If the term is not present at the taxonomy database an error is thrown.

At the moment of this writing, These are the taxonomy levels present at the taxonomy database:

C<class>

C<family>

C<forma>

C<genus>

C<infraclass>

C<infraorder>

C<kingdom>

C<order>

C<parvorder>

C<phylum>

C<species>

C<species group>

C<species subgroup>

C<subclass>

C<subfamily>

C<subgenus>

C<subkingdom>

C<suborder>

C<subphylum>

C<subspecies>

C<subtribe>

C<superclass>

C<superfamily>

C<superkingdom>

C<superorder>

C<superphylum>

C<tribe>

C<varietas>

But note that not all of them are present in all taxonomies.

=head2 C<get_level_from_name>

Given a taxonomical name, get its level

```
my $level = $taxDB->get_level_from_name($name);
```

```
=head2 C<get_taxid_from_name>
```

Given a taxonomical name, get its taxId

```
my $id = $taxDB->get_taxid_from_name($name);
```

```
=head1 AUTHOR
```

Miguel Pignatelli Moreno

Any comments should be addressed to: miguel.pignatelli@uv.es

```
=head1 LICENSE
```

Copyright 2008 Miguel Pignatelli, all rights reserved.

This library is free software; you may redistribute it and/or modify it under the same terms as Perl itself.

```
=cut
```

```
use strict;
```

```
use warnings;
```

```
use Carp qw/croak/;
```

```
use Data::Dumper;
```

```
our $VERSION = 0.04;
```

```
use constant FS => '\t\\t';
```

```
use constant RS => '\t\\n';
```

```
our %allowed_levels;
```

```
sub _check_level
```

```
{  
  
    my ($self, $level) = @_;  
  
    croak "Level not defined" unless defined $level;;  
  
    return $allowed_levels{$level};  
  
}
```

```
sub _print_levels
```

```
{  
  
    my ($self) = @_;  
  
    print STDERR "$_\n" for sort keys %allowed_levels;  
  
}
```

```
sub new
```

```
{  
  
    my ($class, %args) = @_;  
  
    my %opts;  
  
    $args{'nodes'} or croak "Need the nodes.dmp file";  
  
    $args{'names'} or croak "Need the names.dmp file";  
  
}
```

```
@opts{qw /nodesFile namesFile/} = @args{qw/nodes names/};
```

```
my $save_mem = $args{'save_mem'} || 0;
```

```
my $dictFile;
```

```
if ($args{'dict'}) {
```

```
    $dictFile = $args{'dict'};
```

```
    croak "\nERROR\n$dictFile: File not found\n" unless -e $dictFile;
```

```
        croak "\nERROR\nThe file containing the gi <-> taxid correspondences  
must be converted to binary format.\nThis will increase dramatically the speed of this  
script.\nTo convert that file to binary use the tax2bin.pl script like:\n\nperl tax2bin.pl  
$dictFile > $dictFile.bin\n\nand use the resulting file instead\n" unless (-B $dictFile);
```

```
}
```

```
$opts{dict} = $dictFile;
```

```
$opts{save_mem} = $save_mem;
```

```
my $self = bless \%opts;
```

```
$self -> _build_taxonomy();
```

```
$self -> _name_nodes();
```

```
$self -> _build_dict() if (defined $dictFile);
```

```
return $self;
```

```
}
```



```
sub get_taxonomy_from_gi
```

```
{
```

```
    my ($self, $gi) = @_;
```

```
    my $taxid = $self->get_taxid ($gi);
```

```
    my @tax = $self->get_taxonomy ($taxid);
```

```
    return @tax;
```

```
}
```

```
sub get_taxid
```

```
{
```

```
    my ($self, $gi) = @_;
```

```
    # return $self->_binary_lookup ($gi);
```

```
    return $self->_direct_lookup ($gi);
```

```
}
```

```
sub get_term_at_level_from_gi
```

```
{
```

```

my ($self, $gi, $level) = @_;;

do {

    print STDERR "Level $level not recognized\nAllowed levels:\n";

    $self->_print_levels;

    croak;

} if (! defined $self->_check_level($level));

my $taxid = $self->get_taxid($gi);

return $self->get_term_at_level($taxid,$level);

}

```

```

sub _build_dict

{

    my ($self) = @_;;

    my $dictFile = $self->{dict};

    my $data;

    open my $gi_FH, '<:raw', $dictFile or croak "$!:$dictFile";

    if ($self->{save_mem}){

        $self->{fh} = $gi_FH;

    } else {

```

```

    sysread( $gi_FH, $data, -s( $dictFile ) ) or croak $!;

    close $gi_FH;

    $self->{dict} = $data;

}

}

sub _build_taxonomy

{

    my ($self) = @_ ;

    my $nodesFile = $self->{nodesFile};

    my $tax;

    if ((UNIVERSAL::isa($nodesFile, 'GLOB')) or (ref \$nodesFile eq 'GLOB')) {

        $tax = $nodesFile;

    } else {

        open $tax, "<", $nodesFile or croak $!;

    }

    while (<$tax>){

        chomp;

        _create_node (_parse_tax_rec($_));

```

```
}
```

```
close $tax;
```

```
}
```

```
{
```

```
my %nodes;
```

```
my %synonyms;          # JT
```

```
sub _create_node
```

```
{
```

```
my ($node,$parent,$level) = @_;
```

```
$allowed_levels{$level} = 1 if (! defined $allowed_levels{$level});
```

```
@{$nodes{$node}}{qw/parent level/} = ($parent,$level);
```

```
}
```

```
sub _name_nodes
```

```
{
```

```
my ($self) = @_;
```

```
my $namesFile = $self->{namesFile};
```

```

my $nodesNames;

{
    if ((UNIVERSAL::isa($namesFile, 'GLOB')) or (ref \$namesFile eq 'GLOB'))

        $nodesNames = $namesFile;

    } else {

        open $nodesNames, "<", $namesFile or croak $!;

    }

    while (<$nodesNames>){

        chomp;

        my ($taxId,$taxName,$comment) = _process_tax_name ($_);

        $synonyms{$taxName}=$taxId;          # JT

        if ($comment eq "scientific name"){

            ${$nodes{$taxId}}{name} = $taxName;

        }

    }

    close $nodesNames;

}

sub get_term_at_level

```

```

{

    my ($self,$taxid,$level) = @_;

    do {

        print STDERR "Level $level not recognized\nAllowed levels:\n";

        $self->_print_levels;

        croak;

    } if (! defined $self->_check_level($level));

    return "" unless defined ${$nodes{$taxid}}{name};

    while (${ $nodes{$taxid}}{name} ne "root"){

        return ${$nodes{$taxid}}{name} if (${ $nodes{$taxid}}{level} eq
$level);

        $taxid = ${$nodes{$taxid}}{parent};

    }

    return "undef";

}

```

```

sub get_taxonomy

```

```

{

    my ($self, $taxid) = @_;

    return "" unless defined ${$nodes{$taxid}}{name};

```

```

my @taxonomy;

while (${$nodes{$taxid}}{name} ne "root"){

    push @taxonomy, ${$nodes{$taxid}}{name};

    $taxid = ${$nodes{$taxid}}{parent};

}

return reverse do{pop @taxonomy;@taxonomy};

}

```

```

sub get_taxonomy_with_levels

```

```

{

    my ($self,$taxid) = @_;

    return "" unless defined ${$nodes{$taxid}}{name};

    my @taxonomy;

    while (${$nodes{$taxid}}{name} ne "root"){

        push    @taxonomy,    [${$nodes{$taxid}}{name},${$nodes{$taxid}}
{level}]];

        $taxid = ${$nodes{$taxid}}{parent};

    }

    return reverse do{pop @taxonomy;@taxonomy};

}

```

```
}
```

```
sub get_taxonomy_with_levels_from_gi
```

```
{    #JT
```

```
    my ($self, $gi) = @_;
```

```
    my $taxid = $self->get_taxid ($gi);
```

```
    return "" unless defined ${$nodes{$taxid}}{name};
```

```
    my @taxonomy;
```

```
    while (${ $nodes{$taxid} }{name} ne "root"){
```

```
        push @taxonomy, [ ${ $nodes{$taxid} }{name}, ${ $nodes{$taxid} }  
{level}];
```

```
        $taxid = ${ $nodes{$taxid} }{parent};
```

```
    }
```

```
    return reverse do{pop @taxonomy;@taxonomy};
```

```
}
```

```
sub get_level_from_name
```

```
{
```

```
    my ($self,$name) = @_;
```

```
    for (keys %nodes) {
```



```
    return ${$nodes{$_}}{level} if (${ $nodes{$_}}{name} eq $name);

}

return undef;

}
```

```
sub get_taxid_from_name          # JT

{

    my ($self,$name) = @_;

    my $tid=$synonyms{$name};

    return undef if(!$tid);

    my @tax = $self->get_taxonomy($tid);

    my $sk = shift @tax;

    if ($sk ne "Eukaryota") { return $tid; } else { return undef; }

}
```

```
sub get_taxid_from_name_old

{

    my ($self,$name) = @_;

    for (keys %nodes){
```

```
if (${$nodes{$_}}{name} eq $name){  
  
    my @tax = $self->get_taxonomy($_);  
  
    my $sk = shift @tax;  
  
    return $_ if ($sk ne "Eukaryota");  
  
}  
  
}  
  
return undef;  
  
}
```

```
sub get_taxonomy_from_name
```

```
{  
  
    my ($self,$name) = @_;  
  
    my $taxid = $self->get_taxid_from_name($name);  
  
    return $self->get_taxonomy($taxid);  
  
}  
  
}
```

```
sub _parse_tax_rec
```

```
{  
  
    my $line = shift @_;  
  
    return (split FS,$line)[0,1,2];  
  
}
```

```
sub _process_tax_name  
  
{  
  
    my $line = shift @_;  
  
    my @fields = split FS, $line;  
  
    $fields[3] =~ s/\t\|$/;/;  
  
    return ($fields[0],$fields[1],$fields[3]);  
  
}
```

```
sub _binary_lookup {  
  
    my ($self, $gi) = @_;  
  
    my $target = pack 'N', $gi;  
  
    my( $left, $right ) = ( 0, ( length( $self->{dict} ) ) / 8 );  
  
    while( $left < $right ) {
```

```

my $mid = int( ( $left + $right ) / 2 );

my $key = substr $self->{dict}, $mid * 8, 4;

if( $key lt $target ) {

    $left = $mid + 1;

}

elsif( $key gt $target ) {

    $right = $mid;

}

elsif( $key eq $target ) {

    my( $key, $val ) = unpack 'NN', substr $self->{dict}, $mid * 8, 8;

    return $val;

}

}

}

}

sub _direct_lookup {

    my ($self,$gi) = @_ ;

    if ($self->{save_mem}){

        my $taxid;

```

```

sysseek ($self->{fh},$gi*4,0);

sysread($self->{fh},$taxid,4,);

return (unpack "N",$taxid);

} else {

    return (unpack "N",substr($self->{dict},$gi*4,4));
}
}
1;

```

+ANEXO II (Programas propios):

-microparser.pl:

```

#!/usr/bin/perl -w

use DBI;
use strict;
use warnings;

my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );

unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}

#conexion a la base de datos

#Vamos a ejecutar la sentencia para preparar la introduccion a la base de datos
my $sthmicro = $dbh->prepare("INSERT INTO micro(Titulo, Autores, Pmid,
Fecha, Isolation_source, Gen, Secuencia, Metadatos) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");

unless ( defined ($sthmicro) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}

##### CREAR EL FASTA PARA HACE EL BLAST
open (FASTA, '+>fasta.fa')||die "ERROR: no se puede leer o crear el archivo
fasta\n";

open (TITULO, '+>titaut.txt')||die "ERROR: no se puede leer o crear el archivo

```

```

titulo\n";
    open (PACK, '+>PACK.txt')||die "ERROR: no se puede leer o crear el archivo
titulo\n";
    open (FECH, '+>fechpmid.txt')||die "ERROR: no se puede leer o crear el
archivo titulo\n";
    open (ISOLA, '+>isolation.txt')||die "ERROR: no se puede leer o crear el
archivo datos\n";
    #####

my $errflag = 0;
my $file = "";
my $arch = "";

##### LEER LOS ARCHIVOS DE
UNO EN UNO
my $i = 1;
while ($i){
    $file = "gbenv$i.seq";

##### Bucle para cada archivo a
parsear

    open( MICRO, $file )||die "ERROR: no se puede leer o crear el archivo
$file \n" && exit;
    print "Analizando archivo $file\n";
    # Procesamiento fichero

        #limpiar variables
        my $line    = "";
        my $letters = "";
        my $nombre  = "";
        my $titulo  = "";
        my $journal1 = "";
        my $autores = "";
        my $pmid    = "";
        my $fecha   = "";
        my $isolation = "";
        my $gen      = "";
        my $seq      = "";
        my $se1      = "";

```

```

my $tit      = "";
my $title    = "";
my $hitz     = "";
my $autore   = "";
my $_        = "";
my $num      = "";
my $sekuen   = "";
my $DNA      = "";
my $sequ     = "";
my $journal  = "";
my $fine     = "";
my $t1       = "";
my $country  = "";
my $host     = "";
my $coor     = "";
my $metadata = "";
my $au       = "";
my $titu     = "";
my $zen      = 0;
my $titaut   = "";

```

```

foreach (<MICRO>) {
    my $line = $_;
    chomp($line);

```

#expresiones regulares

```

##          TITULO          DEL          ARTICULO
#####
#
    if ($line =~ /TITLE/){
        $title = 1;
    }
    if ($line =~ /JOURNAL\s+\(in\).+/{
        if (!$titulo&& !$title){
            $title = 1;
            $fine = 1;
        }
    }
}

```

```

if ($line =~ /JOURNAL\s+\w+/{
    $journal1 = 1;
}
if (!$journal1){
    if ($title){
        $letters = $_;
        chomp ($letters);
    }
    if ($_ =~ /JOURNAL\s+[a-zA-Z0-9]+/{
        $title = 0;
    }
    $titulo .= $letters;
    $titulo =~ s/TITLE//g;
    if ($titulo =~ /JOURNAL\s+\(in\)(.+)\..+/{
        $titulo = $1;
    }
    if ($titulo =~ /(.)\s+d{4}.+/{
        $titulo = $1;
    }
    $titulo =~ s/JOURNAL\s+\(in\)\s//g;
    if ($titulo =~ /.+(\(EDs.+s\d+\-?\s(.+);.+)\).+/{
        $titulo = $1;
    }
    if ($titulo =~ /(.)\s+AUTHORS.+/{
        $titulo = $1;
    }
    $titulo =~ s/JOURNAL.+//g;
    $titulo =~ s/\s{7}//g;
    $titulo =~ s/\s{4}//g;
    if ($fine){
        if ($titulo =~ /(.)[^sp|spp]\.[^d+]\s+(.+)/g)
        {
            $t1 = $1;
            $titulo = $t1;
        }
        if ($titulo =~ /(.)\s+Direct\s+Submission/{
            $titulo = $1;
        }
    }
}

```


}

}

```
## AUTORES DEL ARTICULO
#####
#
```

```
if ( $line =~ /AUTHORS/ ) {
    $autore = 1;
}
if ($line =~ /TITLE|JOURNAL/){
    $journal = 1;
}
if (!$journal){
    if ($autore){
        $nombre = $_;
        chomp ($nombre);
    }
    if ($_ =~ /TITLE/){
        $autore = 0;
    }
    $autores .= $nombre;
    $autores =~ s/AUTHORS//g;
    $autores =~ s/JOURNAL.+//g;
    $autores =~ s/{7}//g;
    $autores =~ s/{4}//g;
}
}
```

```
## EL NUMERO DE IDENTIFICADOR
#####
```

```
if ( $line =~ /LOCUS\s+(\w{2,12})\s+.+/ ) {
    $pmid = $1;
}
}
```

```
# LA FECHA DE PUBLICACION
#####
```

```
if ( $line =~ /JOURNAL\s+Submitted\s+((\d+-\w+-\d+)\.
+ / ) {
    $fecha = $1;
}
```

```

    }

    # EL MEDIO DONDE SE RECOGIO LA MUESTRA
    #####
    elsif ( $line =~ /\s+\\isolation_source="(.)"$/ ) {
        $isolation = $1;
    }

    # EL NOMBRE DEL GEN
    #####
    #####
    if ( $line =~ /\s+\\product="(.)"/ ) {
        $gen = $1;
    }

    ## METADATOS
    #####
    #
    if ($line =~ /\s+\\country="(.)"$/){
        $country = $1;
    }
    elsif ($line =~ /\s+\\host="(.)"$/){
        $host = $1;
    }
    elsif ($line =~ /\s+\\lat_lon="(.)"$/){
        $coor = $1;
    }

    ## Pais;hospedador;coordenadas ## si las hay
    $metadata= "Lugar:$country;Hospedador:
    $host;Coordenadas:$coor";

    # LA SECUENCIA
    #####
    #####
    if ( $line =~ /^ORIGIN/ ) { # REVISARLO
        $se1 = 1;
    }
    if ($se1){

        $sekuen = $_;
    }

```

```

        chomp ($sekuen);
        $seq .= $sekuen;
    if ($_ =~ /\V\//) {
        $se1 = 0;
        $seq =~ s/ORIGIN//g;
        $seq =~ s/\d//g;
        $seq =~ s/\s//g;
        $seq =~ s/\V\///g;
    }
}

```

if (\$line =~ /^V\//){ #cuando acabe cada entrada
(con //) guarda los datos en la BD

```

        #Solo las entradas con 16S rRNA
        if ($titulo !~ /.+/{
            $titulo = ".";
        }
        $titaut = "$titulo|$autores";
        #print $titaut;
        if ( ($gen =~ /(16S\s*.*|
        ([Ss]mall\s+subunit\s+ribosomal\s+RNA)/) ){
            if (length $seq > 200 && length $seq < 1800){
                print FASTA ">$pmid\n$seq\n\n";
                print TITULO "$titaut\n";
                print PACK "$pmid|$titulo|$autores\n";
                if ($fecha=~/\d{2}\-\w{3}\-(\d{4})/){
                    $ano=$1;
                }
                print FECH "$ano|$pmid\n";
                print ISOLA "$isolation\n";
                #Titulo, Autores, Pmid, Fecha,
        Isolation_source, Gen, Secuencia, Metadatos

```

```

        if ( !$sthmicro->execute($titulo,
        $autores, $pmid, $fecha, $isolation, $gen, $seq, $metadata)) {
            warn "error al insertar: " .
        $DBI::errstr;

            $errflag = 1;
            exit;

```

```

    }

    if ( !$errflag ) {
        $dbh->commit();
    }
    else {
        $dbh->rollback();
    }

}
}else {$gen = "";
}
#limpiar variables
$titulo    = "";
$autores   = "";
$pmid      = "";
$fecha     = "";
$isolation = "";
$gen       = "";
$seq       = "";
$tit       = "";
$se1       = "";
$letters   = "";
$title     = "";
$hitz      = "";
$autore    = "";
$nombre    = "";
$journal   = "";
$journal1  = "";
$fine      = "";
$t1        = "";
$host      = "";
$coor      = "";
$country   = "";
$metadata  = "";
$titu      = "";
$au        = "";
$zen       = 0;

```

```

        $titaut = "";
    }
}    #cierra las expresiones regulares del parseador

close(MICRO);

$i++;
}
##### Fin del bucle de cada archivo
#Cerrar todas las transacciones con las tablas
$sthmicro->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

#Cerrar la coneccion con la base de datos y el archivo
$dbh->disconnect() || warn " Fallo al desconectar . Error :
$DBI::errstr \n ";
close(ISOLA);
close(FASTA);
close(TITULO);
close(PACK);
close(FECH);
exit;

```

-muestratabla.pl

```

#!/usr/bin/perl -w

use DBI;
use strict;
use warnings;

my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );
unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}
#conexion a la base de datos
#Vamos a ejecutar la sentencia para preparar la introduccion a la base de datos
my $sthmicro = $dbh->prepare("INSERT INTO micro(Titulo, Autores, Pmid,
Fecha, Isolation_source, Gen, Secuencia, Metadatos) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");

```

```
unless ( defined ($sthmicro) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}
```

```
##### CREAR EL FASTA PARA HACE EL BLAST
open (FASTA, '+>fasta.fa')||die "ERROR: no se puede leer o crear el archivo
fasta\n";
open (TITULO, '+>titaut.txt')||die "ERROR: no se puede leer o crear el archivo
titulo\n";
open (PACK, '+>PACK.txt')||die "ERROR: no se puede leer o crear el archivo
titulo\n";
open (FECH, '+>fechpmid.txt')||die "ERROR: no se puede leer o crear el
archivo titulo\n";
open (ISOLA, '+>isolation.txt')||die "ERROR: no se puede leer o crear el
archivo datos\n";
#####
```

```
my $errflag = 0;
my $file = "";
my $arch = "";
```

```
##### LEER LOS ARCHIVOS DE
UNO EN UNO
my $i = 1;
while ($i){
    $file = "gbenv$i.seq";
```

```
##### Bucle para cada archivo a
parsear
```

```
open( MICRO, $file )||die "ERROR: no se puede leer o crear el archivo
$file \n" && exit;
print "Analizando archivo $file\n";
# Procesamiento fichero
```

```
#limpiar variables
my $line    = "";
my $letters = "";
my $nombre  = "";
my $titulo  = "";
my $journal1 = "";
my $autores = "";
```

```

my $pmid      = "";
my $fecha     = "";
my $isolation = "";
my $gen       = "";
my $seq       = "";
my $se1       = "";
my $tit       = "";
my $title     = "";
my $hitz      = "";
my $autore    = "";
my $_         = "";
my $num       = "";
my $sekuen    = "";
my $DNA       = "";
my $sequ      = "";
my $journal   = "";
my $fine      = "";
my $t1        = "";
my $country   = "";
my $host      = "";
my $coor      = "";
my $metadata  = "";
my $au        = "";
my $titu      = "";
my $zen       = 0;
my $titaut    = "";

```

```

foreach (<MICRO>) {
    my $line = $_;
    chomp($line);

```

```

    #expresiones regulares

```

```

                                ##          TITULO          DEL          ARTICULO
#####
#
                                if ($line =~ /TITLE/){
                                    $title = 1;
                                }

```

```

if ($line =~ /JOURNAL\s+\(in\).+/{
    if (!$titulo&& !$title){
        $title = 1;
        $fine = 1;
    }
}
if ($line =~ /JOURNAL\s+\w+/{
    $journal1 = 1;
}
if (!$journal1){
    if ($title){
        $letters = $_;
        chomp ($letters);
    }
    if ($_ =~ /JOURNAL\s+[a-zA-Z0-9]+/{
        $title = 0;
    }
    $titulo .= $letters;
    $titulo =~ s/TITLE//g;
    if ($titulo =~ /JOURNAL\s+\(in\)(.+)\..+/{
        $titulo = $1;
    }
    if ($titulo =~ /(.)\s+d{4}.+/{
        $titulo = $1;
    }
    $titulo =~ s/JOURNAL\s+\(in\)\s//g;
    if ($titulo =~ /.+(\(EDs.+s\d+-?\s(.+);.+)\).+/{
        $titulo = $1;
    }
    if ($titulo =~ /(.)\s+AUTHORS.+/{
        $titulo = $1;
    }
    $titulo =~ s/JOURNAL.+//g;
    $titulo =~ s/\s{7}//g;
    $titulo =~ s/\s{4}//g;
    if ($fine){
        if ($titulo =~ /(.)[^sp|spp]\.[^d+]\s+(.+)/g)

```

```
{
```



```

        $t1 = $1;
        $titulo = $t1;
    }
    if ($titulo =~ /(.)\s+Direct\s+Submission/){
        $titulo = $1;
    }
}

}

```

```

##          AUTORES          DEL          ARTICULO
#####
#

```

```

if ( $line =~ /AUTHORS/ ) {
    $autore = 1;
}
if ($line =~ /TITLE|JOURNAL/){
    $journal = 1;
}
if (!$journal){
    if ($autore){
        $nombre = $_;
        chomp ($nombre);
    }
    if ($_ =~ /TITLE/){
        $autore = 0;
    }
    $autores .= $nombre;
    $autores =~ s/AUTHORS//g;
    $autores =~ s/JOURNAL.+//g;
    $autores =~ s/\s{7}//g;
    $autores =~ s/\s{4}//g;
}
}

```

```

##          EL          NUMERO          DE          IDENTIFICADOR
#####

```

```

if ( $line =~ /LOCUS\s+(\w{2,12})\s+.+/ ) {
    $pmid = $1;
}

```

```

    }

#          LA          FECHA          DE          PUBLICACION
#####
    if ( $line =~ /JOURNAL\s+Submitted\s+\((\d+-\w+-\d+)\).
+ / ) {

        $fecha = $1;

    }

#  EL  MEDIO  DONDE  SE  RECOGIO  LA  MUESTRA
#####
    elsif ( $line =~ /\s+V/isolation_source="(.)"$/ ) {
        $isolation = $1;
    }

#          EL          NOMBRE          DEL          GEN
#####
#####

    if ( $line =~ /\s+V/product="(.)"/ ) {
        $gen = $1;
    }

##                                     METADATOS
#####
#

    if ($line =~ /\s+V/country="(.)"$/){
        $country = $1;
    }
    elsif ($line =~ /\s+V/host="(.)"$/){
        $host = $1;
    }
    elsif ($line =~ /\s+V/lat_lon="(.)"$/){
        $coor = $1;
    }

    ## Pais;hospedador;coordenadas ## si las hay
    $metadata=                                "Lugar:$country;Hospedador:
$host;Coordenadas:$coor";

#          LA          SECUENCIA
#####

```

#####

```
if ( $line =~ /^ORIGIN/ ) { # REVISARLO
```

```
    $se1 = 1;
```

```
}
```

```
if ($se1){
```

```
    $sekuen = $_;
```

```
    chomp ($sekuen);
```

```
    $seq .= $sekuen;
```

```
    if ($_ =~ /\V\//) {
```

```
        $se1 = 0;
```

```
        $seq =~ s/ORIGIN//g;
```

```
        $seq =~ s/\d//g;
```

```
        $seq =~ s/\s//g;
```

```
        $seq =~ s/\V\//g;
```

```
    }
```

```
}
```

if (\$line =~ /\V\//) { #cuando acabe cada entrada
(con //) guarda los datos en la BD

```
#Solo las entradas con 16S rRNA
```

```
if ($titulo !~ /.+/){
```

```
    $titulo = ".";
```

```
}
```

```
$titaut = "$titulo|$autores";
```

```
#print $titaut;
```

if (\$gen =~ /(16S\s*.*)|
([Ss]mall\s+subunit\s+ribosomal\s+RNA)/){

```
    if (length $seq > 200 && length $seq < 1800){
```

```
        print FASTA ">$pamid\n$seq\n\n";
```

```
        print TITULO "$titaut\n";
```

```
        print PACK "$pamid|$titulo|$autores\n";
```

```
        if ($fecha =~ /\d{2}\-\w{3}\-(\d{4})/){
```

```
            $ano=$1;
```

```
        }
```

```
        print FECH "$ano|$pamid\n";
```

```
        print ISOLA "$isolation\n";
```

```
#Titulo, Autores, Pmid, Fecha,
```

Isolation_source, Gen, Secuencia, Metadatos

```
if ( !$sthmicro->execute($titulo,
$autores, $pmid, $fecha, $isolation, $gen, $seq, $metadata)) {
    warn "error al insertar: " .
$DBI::errstr;

    $errflag = 1;
    exit;
}

if ( !$errflag ) {
    $dbh->commit();
}
else {
    $dbh->rollback();
}

}
}else {$gen = "";
}
#limpiar variables
$titulo = "";
$autores = "";
$pmid = "";
$fecha = "";
$isolation = "";
$gen = "";
$seq = "";
$tit = "";
$se1 = "";
$letters = "";
$title = "";
$hitz = "";
$autore = "";
$nombre = "";
$journal = "";
$journal1 = "";
$fine = "";
$t1 = ";
```

```

        $host = "";
        $coor = "";
        $country = "";
        $metadata = "";
        $titu = "";
        $au = "";
        $zen = 0;
        $titaut = "";
    }
} #cierra las expresiones regulares del parseador

close(MICRO);

$i++;
}
##### Fin del bucle de cada archivo
#Cerrar todas las transacciones con las tablas
$sthmicro->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

#Cerrar la coneccion con la base de datos y el archivo
$dbh->disconnect() || warn " Fallo al desconectar . Error :
$DBI::errstr \n ";
close(ISOLA);
close(FASTA);
close(TITULO);
close(PACK);
close(FECH);
exit;

```

-metada_ambiente.pl:

```

#!/usr/bin/perl -w

use DBI;
use strict;
use warnings;
use Text::LevenshteinXS qw (distance);

```

```
# ASIGNA UN NUMERO IDENTIFICADRO DE AMBIENTE A CADA GRUPO DE
MUESTRAS QUE TENGAN EL MISMO ISOLATION SOURCE Y METADATOS
```

```
# CORREGIR
```

```
my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );
```

```
unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}
```

```
#entre todos los titulos
```

```
open( UNIQ, 'uniquiso.txt' );          #archivo todo corregido sin redundancias
```

```
if ( !open( UNIQ, 'uniquiso.txt' ) ) {die "Error al abrir el fichero
'uniquiso.tx'";}
```

```
my @uniq= <UNIQ>;
```

```
chomp(@uniq);
```

```
### Preparar la tabla de datos para la tabla de archivos purgados
```

```
my $sthpmid = $dbh->prepare("SELECT Pmid, Metadatos FROM micro WHERE
Isolation_source=? " );
```

```
unless (defined($sthpmid)){die "no se pueden preparar las datos para
insertarlos en las tablas\n";}
```

```
my $sthambiente = $dbh->prepare("INSERT INTO ambiente(Pmid,
ID_ambiente, Nombre_tipo, metadata) VALUES (?, ?, ?, ?)");
```

```
unless ( defined ($sthambiente) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}
```

```
my $errflag = 0;
```

```
my $pm="";
```

```
my $line = "";
```

```
my $me = "";
```

```
my @ea = ();
```

```
my $pmid="";
```

```
my $a="";
```

```
my $e="";
```

```
my $i="";
```

```
my $u="";
```

```
my $meta="";
```

```

$i=0;
foreach $a(@uniq){
    $i++;
    unless ( $sthpmid->execute($a) ) {die "Se ha producido un problema al
conectar con la base de datos: " . $DBI::errstr unless ( defined($dbh) );}
    if ($sthpmid->rows == 0) {
        print "NO MATCHES FOR $a.\n\n";
    }
    else {
        while ($pmid, $meta = $sthpmid->fetchrow_array()){
            $e= "$pmid|$meta";
            push(@ea, $e);
        }
        foreach $u(@ea){
            if ($u =~ /(.)\|(.)/){
                $pm=$1;
                $me=$2;
            }

        }
        #print "$pm, $i, $a, $me\n";
        if ( !$sthambiente->execute($pm, $i, $a, $me)) {
            warn "error al insertar: " . $DBI::errstr;
            $errflag = 1;
            exit;
        }
        if ( !$errflag ) {
            $dbh->commit();
        }
        else {
            $dbh->rollback();
        }
    }
}

```

Fin del bucle de cada archivo

```

#Cerrar todas las transacciones con las tablas
open( MUES, '+>id_ambiente.txt' ); #archivo para guardar el ultimo numero
de muestra

if ( !open( MUES, 'id_ambiente.txt' ) ) {die "Error al abrir el fichero
'id_ambiente.txt'";}

print MUES "$i\n";
close(MUES);

$sthambiente->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

$sthpmid->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

#Cerrar la coneccion con la base de datos y el archivo
$dbh->disconnect() || warn " Fallo al desconectar . Error : $DBI::errstr \n ";
close(UNIQ);
exit;

```

-idmuestra.pl:

```

#!/usr/bin/perl -w

use DBI;
use strict;
use warnings;
use Text::LevenshteinXS qw (distance);

# ASIGNA UN NUMERO IDENTIFICADRO DE AMBIENTE A CADA GRUPO DE
MUESTRAS QUE TENGAN EL MISMO ISOLATION SOURCE Y METADATOS

# CORREGIR

my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );

unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}

#entre todos los titulos
open( UNIQ, 'uniquiso.txt' );          #archivo todo corregido sin redundancias

```



```

        if ( !open( UNIQ, 'uniquiso.txt' ) ) {die "Error al abrir el fichero
'uniquiso.tx";}

        my @uniq= <UNIQ>;
        chomp(@uniq);

        ### Preparar la tabla de datos para la tabla de archivos purgados

        my $sthpmid = $dbh->prepare("SELECT Pmid, Metadatos FROM micro WHERE
Isolation_source=? " );

        unless (defined($sthpmid)){die "no se pueden preparar las datos para
insertarlos en las tablas\n";}

        my $sthambiente = $dbh->prepare("INSERT INTO ambiente(Pmid,
ID_ambiente, Nombre_tipo, metadata) VALUES (?, ?, ?, ?)");

        unless ( defined ($sthambiente) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}

        my $errflag = 0;
        my $pm="";
        my $line = "";
        my $me = "";
        my @ea = ();
        my $pmid="";
        my $a="";
        my $e="";
        my $i="";
        my $u="";
        my $meta="";

        $i=0;
        foreach $a(@uniq){
            $i++;
            unless ( $sthpmid->execute($a) ) {die "Se ha producido un problema al
conectar con la base de datos: " . $DBI::errstr unless ( defined($dbh) );}
            if ($sthpmid->rows == 0) {
                print "NO MATCHES FOR $a.\n\n";
            }
            else {
                while ($pmid, $meta = $sthpmid->fetchrow_array()){
                    $e= "$pmid|$meta";

```

```

        push(@ea, $e);
    }
    foreach $u(@ea){
        if ($u =~ /(.)\|(.)/){
            $pm=$1;
            $me=$2;
        }

    }

    #print "$pm, $i, $a, $me\n";
    if ( !$sthambiente->execute($pm, $i, $a, $me)) {
        warn "error al insertar: " . $DBI::errstr;
        $errflag = 1;
        exit;
    }
    if ( !$errflag ) {
        $dbh->commit();
    }
    else {
        $dbh->rollback();
    }
}

}

```

Fin del bucle de cada archivo

#Cerrar todas las transacciones con las tablas

open(MUES, '+>id_ambiente.txt'); #archivo para guardar el ultimo numero de muestra

if (!open(MUES, 'id_ambiente.txt')) {die "Error al abrir el fichero 'id_ambiente.txt'";}

print MUES "\$i\n";

close(MUES);

\$sthambiente->finish() unless (\$DBI::err);

warn "Error de consulta: " . \$DBI::errstr if (\$DBI::err);

\$sthpmid->finish() unless (\$DBI::err);

warn "Error de consulta: " . \$DBI::errstr if (\$DBI::err);

```
#Cerrar la coneccion con la base de datos y el archivo
$dbh->disconnect() || warn " Fallo al desconectar . Error : $DBI::errstr \n ";
close(UNIQ);
exit;
```

-coordinador.pl

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

```
use strict;
```

```
use warnings;
```

```
#####
```

```
# Programa para coordinar el resto de programas
```

```
#####
```

```
#####
```

```
# limpiar variables
```

```
my $a = "";
```

```
my $e = "";
```

```
my $i = 1;
```

```
# El programa podra hacer la carga 'de novo' o actualizar la base de datos
```

```
while ($i==1){
```

```
    print "Quieres actualizar la base de datos de genbank?\nA: Cargar la
base de datos\nB: Actualizar la base de datos\nC: Salir\n";
```

```
    $a= <STDIN>;
```

```
    chomp $a;
```

```
    $a = uc $a;
```

```
    if ($a =~ /B/){          #ACTUALIZAR LA BASE DE DATOS
```

```
        print "Estas seguro de querer actualizar la base de datos?(S/n)\n";
```

```
        $e = <STDIN>;
```

```
        $e = uc $e;
```

```
        if ($e =~ /SI|S/){
```

```
            system ('rm gbenv*.seq');      # Elimina todas las entradas
existentes de la serie gbenv (si existen)
```

```
            system ('wget ftp://ftp.ncbi.nih.gov/genbank/gbenv*'); #
se descarga los datos del genbank
```

```

                                system ('gzip -d gbenv*.seq.gz');      #descomprime  todos
los archivos descargados
                                system ('perl acmicroparser.pl');      # parsea los archivos
del genbank y crea en outfasta2
                                system ('perl idmuestra.pl');      #inserta el id de muestra
para cada caso
                                system ('perl muestratabla.pl'); #inserta los datos referentes
a las muestras en la tabla (muestra)
                                system ('perl acmetadata_ambiente.pl');      #inserta      los
datos referentes al ambiente
                                system ('cat outfasta* > fasta3');
                                system ('cd-hit-est -i fasta3 -o outfasta3 -c 0.97 -M 4000 -T
2 -aL 0.8 -aS 0.4 -l 200 -r 1'); #comando del DC-HIT-EST
                                system ('vm outfasta3 outfasta');      #renombra el fichero
final
                                system ('vm outfasta3.clstr outfasta.clstr'); #renombra el
fichero final
                                system ('perl actualizacion_clusteres.pl');      #insertar
elarchivo del CD_HIT en la BD
                                system ('perl acconteofasta.pl');      #fracciona los fasta
para realizar el el blast
                                system ('perl blastn.pl'); #hace el blast
                                system ('cat *.fas.blastn > fichero.fas.blastn');
#concatena todos los archivos en uno solo
                                system ('perl asigna16S.pl fichero.fas.blastn > besthit.txt');
# crea el archivo de taxones
                                system ('perl blastparser.pl');      #inserta los datos referentes
a los taxones (taxones y especies)
                                print "La base de datos esta actualizada\n";
                                $i = "";
                                }else{
                                $i = 1;
                                }
                                }
                                elsif ($a =~ /A/){ # CARGAR LA BASE DE DATOS
                                print "Estas seguro de querer cargar la base de datos?(S/n)\n";
                                $e = <STDIN>;
                                $e = uc $e;
                                if ($e =~ /SI|S/){
                                system ('rm gbenv*.seq');      # Elimina todas las entradas
existentes de la serie gbenv (si existen)
                                system ('wget ftp://ftp.ncbi.nih.gov/genbank/gbenv*'); #
se descarga los datos del genbank

```

```

                                system ('gzip -d gbenv*.seq.gz');    #descomprime  todos
los archivos descargados
                                system ('perl microparser.pl'); # parsea los archivos del
genbank y los inserta en la BD (micro)
                                system ('perl idmuestra.pl');    # el id de muestra para cada
caso
                                system ('perl muestratabla.pl'); #inserta los datos referentes
a las muestras en la tabla (muestra)
                                system ('perl metada_ambiente.pl'); #inserta los datos
referentes al ambiente en la tabla (ambientes)
                                system ('cd-hit-est -i fasta.fa -o outfasta -c 0.97 -M 4000 -T
2 -aL 0.8 -aS 0.4 -l 200 -r 1'); #comando del DC-HIT-EST
                                system ('perl cdhitparser.pl'); # inserta el archivo del
CD_HIT en la BD (especies y secuencias)
                                system                                                    ('wget
greengenes.lbl.gov/Download/Sequence_Data/Fasta_data_files/current_prokMSA_una
ligned.fasta.gz'); #descarga la BD del greengenes
                                system ('formatdb -i current_prokMSA_unaligned.fasta -p
F'); # da formato al archivo descargado de greengenes
                                system ('perl conteofasta.pl'); #fracciona los fasta para
realizar el el blast
                                system ('perl blastn.pl'); #hace el blast
                                system ('cat *.fas.blastn > fichero.fas.blastn');
#concatena todos los archivos en uno solo
                                system ('perl asigna16S.pl fichero.fas.blastn > besthit.txt');
# crea el archivo de taxones
                                system ('perl blastparser.pl'); #inserta los datos referentes
a los taxones (taxones y especies)
                                print "La base de datos esta cargada\n";
                                $i = "";
                                }else{
                                    $i = 1;
                                }
                            }
                        elsif ($a=~ /C/){ #SALIR DE LA BASE DE DATOS
                            exit;
                        }
                    else { #SALIR DE LA BASE DE DATOS
                        exit;
                    }
                }
            }
        exit;

```

-conteofasta.pl:

```
#!/usr/bin/perl -w
use DBI;
use strict;
use warnings;
# Cuando se caraga la base de datos por primera vez
open( FAST, 'outfasta' );
    if ( !open( FAST, 'outfasta' ) ) {die "Error al abrir el fichero 'outfasta'";}

my $numfa = 0;
my $i = 0;
foreach (<FAST>) {
    open (PART, ">fas$i.txt")||die "ERROR: no se puede leer o crear el
archivo titulo\n";
    my $line = $_;
    chomp($line);
    if ($line =~ /^>/){
        $numfa++;
    }
    print PART $line;
    if ($numfa == 50000){
        if ($line !~ /.+/){
            close(PART);
            $i++;
        }
    }
}
close(FAST);
close(PART);

exit;
```

-cdhitparser.pl:

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

```

use DBI;
#use strict;
use warnings;
#####
# Parseador de los archivos devulestos por el CD-HIT
# Y meterlo en la base de datos

#####
#####

my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );
unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}

#####
open (CDHIT, 'outfasta.clstr')||die "ERROR: no se puede leer o crear el archivo
fasta\n";

#conexion a la base de datos
# Vamos a ejecutar la sentencia para preparar la introducion a la base
de datos

my $sthesp = $dbh->prepare("INSERT INTO especies(ID_cluster,
NUM_seq, ID_seq_representante) VALUES (?, ?, ?)");
unless ( defined ($sthesp) ) {die "no se pueden prepara las datos
para insertarlos en las tablas\n";}

my $sthseq = $dbh->prepare("INSERT INTO
secuencias(ID_cluster, longitud_seq, Pmid, Secuencia) VALUES (?, ?, ?, ?)");
unless ( defined ($sthseq) ) {die "no se pueden prepara las datos
para insertarlos en las tablas\n";}

my $sthsecu = $dbh->prepare("SELECT Secuencia FROM micro
WHERE Pmid=? " );
unless ( defined ($sthsecu) ) {die "no se pueden prepara las datos
para insertarlos en las tablas\n";}

#####
# Preparar las variables a utilizar
my $errflag = 0;
my $cluster = "";

```

```

my $long = "";
my $seq = "";
my @reflong = ();
my @refseq = ();
my $numseq = "";
my $seqrepr = "";
my $i = "";
my $reflong1 = "";
my $u = "";
my $secuencia = "";

```

```

#####

```

```

foreach (<CDHIT>) {

```

```

    my $line = $_;
    chomp($line);

```

```

    #expresiones regulares

```

```

        ## ID_CLUSTER

```

```

    if ($line =~ /^>Cluster\s+(\d+)/){

```

```

        $cluster = $1;

```

```

    }

```

```

    elsif ($line =~ /^[^>]/){

```

```

        if ($line =~ /\d+\s+\d+nt,\s+>(\w+)\.{3}\s+\*/){

```

```

            $seqrepr = $1; ### ID de la secuencia representante del

```

cluster

```

        }

```

```

        if ($line =~ /\d+\s+(\d+)nt,\s+>(\w+)\.{3}\s+\*?.*/){

```

```

            ### longitud de secuencia

```

```

            $long = $1;

```

```

            push( @reflong, $long );

```

```

            ### ID de la secuencia

```

```

            $seq = $2;

```

```

            push( @refseq, $seq );

```

```

        }

```

```

    }

```

```

    if ($line =~ /^>){

```



```

$numseq = scalar@refseq;
$u = 0;
foreach $i(@refseq){
    $reflong1 = $reflong[$u];
    unless ( $sthsecu->execute($i) ) {die "Se ha producido un
problema al conectar con la base de datos: " . $DBI::errstr unless ( defined($dbh) );}
    #imprime los valores que se hayan pedido
    if ($sthsecu->rows == 0) {
        print "NO MATCHES FOR $i.\n\n";
    }
    else {
        $secuencia = $sthsecu->fetchrow_array();
    }
    ## ID_cluster      longitud_seqrefseqsecuencia
    if ( !$sthseq->execute($cluster, $reflong1, $i, $secuencia)) {
        warn "error al insertar: " . $DBI::errstr;
        $errflag = 1;
        exit;
    }

    $u++;
}

## ID_cluster,      NUM_seq,  seqrepr
if ( !$sthseq->execute($cluster, $numseq, $seqrepr)) {
    warn "error al insertar: " . $DBI::errstr;
    $errflag = 1;
    exit;
}

```

limpiar variables

```

$numseq = "";
$cluster = "";
$seqrepr = "";
$sequ = "";
$seq = "";

```

```

        @refseq = ();
        @reflong = ();
        $reflong1 = "";
        $u = "";
        $secuencia = "";
        $sequ1 = "";
    }
}
if ( !$errflag ) {
    $dbh->commit();
}
else {
    $dbh->rollback();
}

##### Fin del bucle de cada archivo
#Cerrar todas las transacciones con las tablas
$sthesp->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

$sthseq->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

$sthsecu->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

#Cerrar la coneccion con la base de datos y el archivo
$dbh->disconnect() || warn " Fallo al desconectar . Error : $DBI::errstr \n ";

close(CDHIT);

exit;

```

-blastparser.pl:

```

#!/usr/bin/perl -w

use DBI;

```

```
use strict;
use warnings;
```

```
my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );
```

```
unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}
```

```
##### ABRIR ARCHIVO
```

```
open (DATOS, 'besthit.txt')||die "ERROR: no se puede leer o crear el archivo
datos de blast\n";
```

```
# Procesamiento fichero
```

```
#conexion a la base de datos
```

```
# Vamos a ejecutar la sentencia para preparar la introducion a la base de
datos
```

```
my $sthtax = $dbh->prepare("INSERT INTO taxones( ID_seq_representante,
superkingdom_besthit, superkingdom_bestAVG, superkingdom_ID, phylum_besthit,
phylum_bestAVG, phylum_ID, class_besthit, class_bestAVG, class_ID, orderbesthit,
orderbestAVG, orderID, family_besthit, family_bestAVG, family_ID, genus_besthit,
genus_bestAVG, genus_ID, species_besthit, species_bestAVG, species_ID) VALUES (?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
```

```
unless ( defined ($sthtax) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}
```

```
my $errflag = 0;
```

```
my $pmid = "";
```

```
my $tipo = "";
```

```
my $nomAVG = "";
```

```
my $nomHIT = "";
```

```
my $score = "";
```

```
my $kingAVG = "";
```

```
my $kingHIT = "";
```

```
my $kingscor = "";
```

```
my $phyAVG = "";
```

```
my $phyHIT = "";
```

```
my $physcor = "";
```

```
my $clasAVG = "";
```

```
my $clasHIT = "";
```

```
my $classcor = "";
my $ordAVG = "";
my $ordHIT = "";
my $ordscor = "";
my $famAVG = "";
my $famHIT = "";
my $famscor = "";
my $genAVG = "";
my $genHIT = "";
my $genscor = "";
my $specAVG = "";
my $specHIT = "";
my $specscor = "";
my $tax = "";
my $sthtaxon = "";
my $tex = "";
```

```
foreach (<DATOS>) {
    my $line = $_;
    chomp($line);

    if ($line =~ /^(.+)\t(.+)\t(.+)\t(.+)/){
        $pmid = $1;
        $tipo = $2;
        $nomAVG = $3;
        $nomHIT = $4;
        $score = $5;
        if ($tipo =~ /superkingdom/ ){
            $kingAVG=$nomAVG;
            $kingHIT=$nomHIT;
            $kingscor=$score;
        }
        elsif ($tipo =~ /phylum/ ){
            $phyAVG=$nomAVG;
            $phyHIT=$nomHIT;
            $physcor=$score;
        }
    }
}
```

```

elseif ($tipo =~ /class/ ){
    $clasAVG=$nomAVG;
    $clasHIT=$nomHIT;
    $classcor=$score;
}
elseif ($tipo =~ /order/ ){
    $ordAVG=$nomAVG;
    $ordHIT=$nomHIT;
    $ordscor=$score;
}
elseif ($tipo =~ /family/ ){
    $famAVG=$nomAVG;
    $famHIT=$nomHIT;
    $famscor=$score;
}
elseif ($tipo =~ /genus/ ){
    $genAVG=$nomAVG;
    $genHIT=$nomHIT;
    $genscor=$score;
}
elseif ($tipo =~ /species/ ){
    $specAVG=$nomAVG;
    $specHIT=$nomHIT;
    $specscor=$score;
}
}

```

```

if ($line !~ /.+/{

```

```

#      print "$pmid, $kingHIT, $kingAVG, $kingscor, $phyHIT, $phyAVG,
$physcor, $clasHIT, $clasAVG, $classcor, $ordHIT, $ordAVG, $ordscor, $famHIT,
$famAVG, $famscor, $genHIT, $genAVG, $genscor, $specHIT, $specAVG,
$specscor\n";

```

```

    if ( $kingAVG=~ /Unresolved/){
        $tax = "$kingAVG(superkingdom)";
    }

```

```

    elseif ($specscor =~ /(97\.)*(98\.)*(99\.)*(100\.)* / && $specAVG!
~/Unresolved/){
        $tax = "$kingAVG(superkingdom);$phyAVG(phylum);
$clasAVG(class);$ordAVG(order);$famAVG(family);$genAVG(genus);
$specAVG(species)";

```

```

    }
    elsif ($genscor =~ /
(94\.)*(95\.)*(96\.)*(97\.)*(98\.)*(99\.)*(100\.)* / && $genAVG !~ /Unresolved/){
        $tax= "$kingAVG(superkingdom);$phyAVG(phylum);
$clasAVG(class);$ordAVG(order);$famAVG(family);$genAVG(genus)";
    }
    else{
        $tax="$kingAVG(superkingdom);$phyAVG(phylum);
$clasAVG(class);$ordAVG(order);$famAVG(family)";
    }
    # print "$tax\n";

    $tex= "UPDATE especies set taxon=? WHERE
ID_seq_representante='$pmid'";
    # print "$tex\n";
    $sthtaxon = $dbh->prepare("$tex");
    unless ( defined ($sthtaxon) ) {die "no se pueden prepara las
datos para insertarlos en las tablas\n";}

    if ( !$sthtaxon->execute($tax)){
        warn "error al insertar: " . $DBI::errstr;
        $errflag = 1;
        exit;
    }
    if ( !$errflag ) {
        $dbh->commit();
    }
    else {
        $dbh->rollback();
    }
    # print "$kingAVG(superkingdom);$phyAVG(phylum);
$clasAVG(class);$ordAVG(order);$famAVG(family);$genAVG(genus);
$specAVG(species)\n";
    if ( !$sthtax->execute($pmid, $kingHIT, $kingAVG, $kingscor,
$phyHIT, $phyAVG, $physcor, $clasHIT, $clasAVG, $classcor, $ordHIT, $ordAVG,
$ordscor, $famHIT, $famAVG, $famscor, $genHIT, $genAVG, $genscor, $specHIT,
$specAVG, $specscor )) {
        warn "error al insertar: " . $DBI::errstr;
        $errflag = 1;
        exit;
    }
}

```

```

        if ( !$errflag ) {
            $dbh->commit();
        }
        else {
            $dbh->rollback();
        }
        $pmid = "";
        $nomAVG = "";
        $nomHIT = "";
        $score = "";
        $kingAVG = "";
        $kingHIT = "";
        $king scor = "";
        $phyAVG = "";
        $phyHIT = "";
        $physcor = "";
        $clasAVG = "";
        $clasHIT = "";
        $classcor = "";
        $ordAVG = "";
        $ordHIT = "";
        $ordscor = "";
        $famAVG = "";
        $famHIT = "";
        $famscor = "";
        $genAVG = "";
        $genHIT = "";
        $genscor = "";
        $specAVG = "";
        $specHIT = "";
        $specscor = "";
        $tax = "";
        $tex = "";
        $sthtaxon->finish() unless ($DBI::err);
        warn "Error de consulta: " . $DBI::errstr if ($DBI::err);
    }
}

```

```
##### Fin del bucle de cada archivo
```

```
#Cerrar todas las transacciones con las tablas
```

```
$sthtax->finish() unless ($DBI::err);
```

```
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);
```

```
#Cerrar la coneccion con la base de datos y el archivo
```

```
$dbh->disconnect() || warn " Fallo al desconectar . Error : $DBI::errstr \n ";
```

```
close(DATOS);
```

```
exit;
```

```
-blastn.pl:
```

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

```
use strict;
```

```
use warnings;
```

```
my $a="";
```

```
my $i = 0;
```

```
my $e=1;
```

```
if ($e ==1){
```

```
    open (FAS, "fas$i.txt")||die ;
```

```
    $a = "blastall -p blastn -i fas$i.txt -d greengenes -o file$i.fas.blast -m 8  
-e 1e-03 -D 200 -a 2";
```

```
    #print "$a\n";
```

```
    system("blastall -p blastn -i fas$i.txt -d greengenes -o file$i.fas.blast -m  
8 -e 1e-03 -D 200 -a 2");
```

```
    $i++;
```

```
}
```

```
close(FAS);
```

```
exit;
```

```
-actualizacion_clusteres.pl:
```

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

```
use DBI;
```



```
use strict;
use warnings;
```

```
## Actualizar los clusteres mediante el cd-hit
```

```
#####
```

```
# Parseador de los archivos devulestos por el CD-HIT
```

```
# Y meterlo en la base de datos
```

```
#####
```

```
#####
```

```
my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );
```

```
unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}
```

```
#####
```

```
open (CDHIT, 'outfasta.clstr')||die "ERROR: no se puede leer o crear el archivo
fasta\n";
```

```
open( FASTA, '+>fastaact' );          #archivo todo corregido sin redundancias
```

```
if ( !open( FASTA, 'fastaact' ) ) {die "Error al abrir el fichero 'fastaact'";}
```

```
#conexion a la base de datos
```

```
# Vamos a ejecutar la sentencia para preparar la introduccion a la base de
datos
```

```
my $sthesp = $dbh->prepare("INSERT INTO especies(ID_cluster,
NUM_seq, ID_seq_representante) VALUES (?, ?, ?)");
```

```
unless ( defined ($sthesp) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}
```

```
my $sthseq = $dbh->prepare("INSERT INTO secuencias(ID_cluster,
longitud_seq, Pmid, Secuencia) VALUES (?, ?, ?, ?)");
```

```
unless ( defined ($sthseq) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}
```

```
my $sthsecu = $dbh->prepare("SELECT Secuencia FROM micro WHERE
Pmid=? " );
```

```
unless ( defined ($sthsecu) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}
```

```
my $sthact = $dbh->prepare("SELECT ID_cluster FROM especies WHERE
ID_seq_representante=? " );
```

```
unless ( defined ($sthact) ) {die "no se pueden prepara las datos para
```

insertarlos en las tablas\n";}

```
#####
```

```
my $numseq = "";
my $cluster = "";
my $seqrepr = "";
my @refseq = ();
my @reflong = ();
my $reflong1 = "";
my $u = "";
my $secuencia = "";
my $i="";
my $errflag=0;
my $seq="";
my $long="";
```

```
#####
```

```
foreach (<CDHIT>) {
```

```
    my $line = $_;
    chomp($line);
```

```
    #expresiones regulares
```

```
        ## ID_CLUSTER
```

```
    if ($line =~ /^>Cluster\s+(\d+)/){
```

```
        $cluster = $1;
```

```
    }
```

```
    elsif ($line =~ /^[^>]/){
```

```
        if ($line =~ /\d+\s+\d+nt,\s+(\w+)\.{3}\s+\*/){
```

```
            $seqrepr = $1; ### ID de la secuencia representante del
```

cluster

```
        }
```

```
        if ($line =~ /\d+\s+(\d+)nt,\s+(\w+)\.{3}\s+\*?.*/){
```

```
            ### longitud de secuencia
```

```
            $long = $1;
```

```
            push( @reflong, $long );
```

```
            ### ID de la secuencia
```

```
            $seq = $2;
```

```

        push( @refseq, $seq );
    }
}

if ($line =~ /^>/){
    $numseq = scalar@refseq;
    my $r=0;
    foreach $i(@refseq){
        unless ( $sthact->execute($i) ) {die "Se ha producido un
problema al conectar con la base de datos: " . $DBI::errstr unless ( defined($dbh) );}
        if ($sthact->rows == 0) {
            unless ( $sthsecu->execute($i) ) {die "Se ha
producido un problema al conectar con la base de datos: " . $DBI::errstr unless
( defined($dbh) );}

            #imprime los valores que se hayan pedido
            if ($sthsecu->rows == 0) {
                print "NO MATCHES FOR $i.\n\n";
            }
            else {
                $secuencia      =      $sthsecu-
>fetchrow_array();
            }
            print FASTA "$i\n$secuencia\n\n";      # Ffasta
de las nuevas secuencias no clusterizadas
        }
    }
    else {
        $cluster = $sthact->fetchrow_array();
        splice (@refseq, $r, 1);
        splice (@reflong, $r, 1);
    }
    $r++;
}
$r=0;

$u = 0;
foreach $i(@refseq){
    $reflong1 = $reflong[$u];
    unless ( $sthsecu->execute($i) ) {die "Se ha producido un
problema al conectar con la base de datos: " . $DBI::errstr unless ( defined($dbh) );}
    #imprime los valores que se hayan pedido

```

```

        if ($sthsecu->rows == 0) {
            print "NO MATCHES FOR $i.\n\n";
        }
    else {
        $secuencia = $sthsecu->fetchrow_array();
    }
    ## ID_cluster      longitud_seqrefseqsecuencia
    if ( !$sthseq->execute($cluster, $reflong1, $i, $secuencia)) {
        warn "error al insertar: " . $DBI::errstr;
        $errflag = 1;
        exit;
    }

    $u++;
}

## ID_cluster,      NUM_seq,  seqrepr
if ( !$sthseq->execute($cluster, $numseq, $seqrepr)) {
    warn "error al insertar: " . $DBI::errstr;
    $errflag = 1;
    exit;
}

# limpiar variables
$numseq = "";
$cluster = "";
$seqrepr = "";
$seq = "";
@refseq = ();
@reflong = ();
$reflong1 = "";
$u = "";
$secuencia = "";
}

}
if ( !$errflag ) {
    $dbh->commit();
}
else {

```

```

$dbh->rollback();
}

##### Fin del bucle de cada archivo
#Cerrar todas las transacciones con las tablas
$sthesp->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

$sthseq->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

$sthsecu->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

$sthact->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

#Cerrar la coneccion con la base de datos y el archivo
$dbh->disconnect()|| warn " Fallo al desconectar . Error : $DBI::errstr \n ";

close(CDHIT);
close(FASTA)
exit;

```

-acmicroparser.pl:

```

#!/usr/bin/perl -w

use DBI;
use strict;
use warnings;

my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );

unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}

#conexion a la base de datos
#Vamos a ejecutar la sentencia para preparar la introduccion a la base de datos
my $sthmicro = $dbh->prepare("INSERT INTO micro(Titulo, Autores, Pmid,

```

```

Fecha, Isolation_source, Gen, Secuencia, Metadatos) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
    unless ( defined ($sthmicro) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}

    my $sthmic = $dbh->prepare("SELECT Titulo, Autores, Fecha FROM micro
WHERE Pmid=? ");

    unless ( defined ($sthmic) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}

    my $sthupd = "";
    ##### CREAR EL FASTA PARA HACE EL BLAST
    open (FASTA, '+>outfasta2')||die "ERROR: no se puede leer o crear el archivo
fasta\n";
    open (TITULO, '+>titaut.txt')||die "ERROR: no se puede leer o crear el archivo
titulo\n";
    open (PACK, '+>PACK.txt')||die "ERROR: no se puede leer o crear el archivo
titulo\n";
    open (FECH, '+>fechpmid.txt')||die "ERROR: no se puede leer o crear el
archivo titulo\n";
    open (ISOLA, '+>isolation.txt')||die "ERROR: no se puede leer o crear el
archivo datos\n";
    #####

    my $errflag = 0;
    my $file = "";
    my $arch = "";

    ##### LEER LOS ARCHIVOS DE
UNO EN UNO
    my $i = 1;
    while ($i){
        $file = "gbenv$i.seq";

        ##### Bucle para cada archivo a
parsear

        open( MICRO, $file )||die "ERROR: no se puede leer el archivo $file \n"
&& exit;
        print "Analizando archivo $file\n";
        # Procesamiento fichero

```

```
#limpiar variables
my $line = "";
my $letters = "";
my $nombre = "";
my $titulo = "";
my $journal1 = "";
my $autores = "";
my $pmid = "";
my $fecha = "";
my $isolation = "";
my $gen = "";
my $seq = "";
my $se1 = "";
my $tit = "";
my $title = "";
my $hitz = "";
my $autore = "";
my $_ = "";
my $num = "";
my $sekuen = "";
my $DNA = "";
my $sequ = "";
my $journal = "";
my $fine = "";
my $t1 = "";
my $country = "";
my $host = "";
my $coor = "";
my $metadata = "";
my $au = "";
my $titu = "";
my $zen = 0;
my $titaut = "";
my $aut = "";
my $fech = "";
my $ano="";
foreach (<MICRO>) {
    my $line = $_;
```

```
chomp($line);
```

```
#expresiones regulares
```

```
##          TITULO          DEL          ARTICULO
#####
#
```

```
if ($line =~ /TITLE/){
```

```
    $title = 1;
```

```
}
```

```
if ($line =~ /JOURNAL\s+\(in\).+){
```

```
    if (!$titulo&& !$title){
```

```
        $title = 1;
```

```
        $fine = 1;
```

```
    }
```

```
}
```

```
if ($line =~ /JOURNAL\s+\w+){
```

```
    $journal1 = 1;
```

```
}
```

```
if (!$journal1){
```

```
    if ($title){
```

```
        $letters = $_;
```

```
        chomp ($letters);
```

```
    }
```

```
    if ($_ =~ /JOURNAL\s+[a-zA-Z0-9]+)/){
```

```
        $title = 0;
```

```
    }
```

```
    $titulo .= $letters;
```

```
    $titulo =~ s/TITLE//g;
```

```
    if ($titulo =~ /JOURNAL\s+\(in\)(.+)\..+){
```

```
        $titulo = $1;
```

```
    }
```

```
    if ($titulo =~ /(.)\s+d{4}.+){
```

```
        $titulo = $1;
```

```
    }
```

```
    $titulo =~ s/JOURNAL\s+\(in\)s//g;
```

```
    if ($titulo =~ /.+(\(EDs.+s\d+?-?s(.+);.+)\).+){
```

```
        $titulo = $1;
```

```
    }
```



```

        if ($titulo =~ /(.)\s+AUTHORS.+){
            $titulo = $1;
        }
        $titulo =~ s/JOURNAL.+//g;
        $titulo =~ s/\s{7}//g;
        $titulo =~ s/\s{4}//g;
        if ($fine){
            if ($titulo =~ /(.)[^\s|spp]\.[^\d+]\s+(.+)/g)
            {
                $t1 = $1;
                $titulo = $t1;
            }
            if ($titulo =~ /(.)\s+Direct\s+Submission/){
                $titulo = $1;
            }
        }
    }
}

```

```

##          AUTORES          DEL          ARTICULO
#####
#

```

```

if ( $line =~ /AUTHORS/ ) {
    $autore = 1;
}
if ($line =~ /TITLE|JOURNAL/){
    $journal = 1;
}
if (!$journal){
    if ($autore){
        $nombre = $_;
        chomp ($nombre);
    }
    if ($_ =~ /TITLE/){
        $autore = 0;
    }
    $autores .= $nombre;
    $autores =~ s/AUTHORS//g;
    $autores =~ s/JOURNAL.+//g;
    $autores =~ s/\s{7}//g;
}

```

```
$autores =~ s/\s{4}//g;
```

```
}
```

```
## EL NUMERO DE IDENTIFICADOR  
#####
```

```
if ( $line =~ /LOCUS\s+(\w{2,12})\s+.+/ ) {
```

```
$pmid = $1;
```

```
}
```

```
# LA FECHA DE PUBLICACION  
#####
```

```
if ( $line =~ /JOURNAL\s+Submitted\s+((\d+-\w+-\d+)\.+/ ) {
```

```
$fecha = $1;
```

```
}
```

```
# EL MEDIO DONDE SE RECOGIO LA MUESTRA  
#####
```

```
elsif ( $line =~ /\s+\\isolation_source="(.)"$/ ) {
```

```
$isolation = $1;
```

```
}
```

```
# EL NOMBRE DEL GEN  
#####  
####
```

```
if ( $line =~ /\s+\\product="(.)"/ ) {
```

```
$gen = $1;
```

```
}
```

```
## METADATOS  
#####  
#
```

```
if ($line =~ /\s+\\country="(.)"$/){
```

```
$country = $1;
```

```
}
```

```
elsif ($line =~ /\s+\\host="(.)"$/){
```

```
$host = $1;
```

```
}
```

```
elsif ($line =~ /\s+\\lat_lon="(.)"$/){
```

```
$coor = $1;
```

```

    }

    ## Pais;hospedador;coordenadas ## si las hay
    $metadata="Lugar:$country;Hospedador:
$host;Coordenadas:$coor";

# LA SECUENCIA
#####
#####

if ( $line =~ /^ORIGIN/ ) { # REVISARLO
    $se1 = 1;
}
if ($se1){

    $sekuen = $_;
    chomp ($sekuen);
    $seq .= $sekuen;
    if ($_ =~ /\V\//) {
        $se1 = 0;
        $seq =~ s/ORIGIN//g;
        $seq =~ s/\d//g;
        $seq =~ s/s//g;
        $seq =~ s/\V\//g;
    }
}

if ( $line =~ /^V\// ) { #cuando acabe cada entrada
(con //) guarda los datos en la BD

    #Solo las entradas con 16S rRNA
    if ($titulo !~ /.+/){
        $titulo = ".";
    }
    $titaut = "$titulo|$autores";
    #print $titaut;
    if ($gen =~ /(16S\s*.*)|
([Ss]mall\s+subunit\s+ribosomal\s+RNA)/){
        if (length $seq > 200 && length $seq < 1800){

            unless ( $sthmic->execute($pmid) ) {die

```

```

        if ($sthmic->rows == 0) {      # En caso de
que no exista la entrada en la BD, la introduce como nueva

            print FASTA ">$pmid\n$seq\n\n";
            print TITULO "$titaut\n";
            print PACK "$pmid|$titulo|$autores\n";
            if ($fecha=~/\d{2}\-\w{3}\-(\d{4})/){
                $ano=$1;
            }
            print FECH "$ano|$pmid\n";
            print ISOLA "$isolation\n";
            #Titulo,      Autores,      Pmid,      Fecha,
Isolation_source, Gen, Secuencia, Metadatos

            if ( !$sthmicro->execute($titulo,
$autores, $pmid, $fecha, $isolation, $gen, $seq, $metadata)) {
                warn "error al insertar: " .
$DBI::errstr;

                $errflag = 1;
                exit;
            }
            if ( !$errflag ) {
                $dbh->commit();
            }
            else {
                $dbh->rollback();
            }
        }
        else { #En caso de que la entrada exista,
comprueba la existencia de alguna actualización

            while ( ($tit, $aut, $fech) = $sthmic-
>fetchrow_array() ) {

                if ($tit ne $titulo || $aut ne
$autores || $fech ne $fecha){

                    $sthupd = $dbh-
>prepare("UPDATE micro set Titulo=$tit, Autores=$aut, Fecha=$fech WHERE
Pmid=$pmid ");

                    unless ( defined ($sthupd) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}

                }
            }
        }
    }
}

```

```

        }
    }else {$gen = "";
    }
    #limpiar variables
    $titulo = "";
    $autores = "";
    $pmid = "";
    $fecha = "";
    $isolation = "";
    $gen = "";
    $seq = "";
    $tit = "";
    $se1 = "";
    $letters = "";
    $title = "";
    $hitz = "";
    $autore = "";
    $nombre = "";
    $journal = "";
    $journal1 = "";
    $fine = "";
    $t1 = "";
    $host = "";
    $coor = "";
    $country = "";
    $metadata = "";
    $titu = "";
    $au = "";
    $zen = 0;
    $titaut = "";
    }
    }
    #cierra las expresiones regulares del parseador
    close(MICRO);
    $i++;
}
##### Fin del bucle de cada archivo

```

```

#Cerrar todas las transacciones con las tablas
$sthmicro->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);
$sthmic->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);
$sthupd->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);
#Cerrar la coneccion con la base de datos y el archivo
$dbh->disconnect() || warn " Fallo al desconectar . Error :
$DBI::errstr \n ";

```

```

close(FASTA);
close(TITULO);
close(PACK);
close(FECH);
close(ISOLA);
exit;

```

-acmetadata_ambiente.pl:

```

#!/usr/bin/perl -w

```

```

use DBI;
use strict;
use warnings;
use Text::LevenshteinXS qw (distance);

```

ASIGNA UN NUMERO IDENTIFICADRO DE AMBIENTE A CADA GRUPO DE MUESTRAS QUE TENGAN EL MISMO ISOLATION SOURCE Y METADATOS

```

# CORREGIR

```

```

my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );

unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}

```

```

#entre todos los titulos

```

```

open( UNIQ, 'uniquiso.txt' );          #archivo todo corregido sin redundancias

```

```

        if ( !open( UNIQ, 'uniqiso.txt' ) ) {die "Error al abrir el fichero
'uniqiso.tx'";}
        my @uniq= <UNIQ>;
        chomp(@uniq);
        open( UNIQ, 'id_ambiente.txt' );          #archivo todo corregido sin
redundancias
        if ( !open( UNIQ, 'id_ambiente.txt' ) ) {die "Error al abrir el fichero
'id_ambiente.txt'";}
        my @amb= <UNIQ>;
        chomp(@amb);
        ### Preparar la tabla de datos para la tabla de archivos purgados

        my $sthpmid = $dbh->prepare("SELECT Pmid, Metadatos FROM micro WHERE
Isolation_source=? ");
        unless (defined($sthpmid)){die "no se pueden preparar las datos para
insertarlos en las tablas\n";}

        my $sthambiente = $dbh->prepare("INSERT INTO ambiente(Pmid,
ID_ambiente, Nombre_tipo, metadata) VALUES (?, ?, ?, ?)");
        unless ( defined ($sthambiente) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}

        my $errflag = 0;
        my $pm="";
        my $line = "";
        my $me = "";
        my @ea = ();
        my $pmid="";
        my $a="";
        my $e="";
        my $i="";
        my $u="";
        my $meta="";
        foreach $i(@amb){
            foreach $a(@uniq){
                $i++;
                unless ( $sthpmid->execute($a) ) {die "Se ha producido un
problema al conectar con la base de datos: " . $DBI::errstr unless ( defined($dbh) );}
                if ($sthpmid->rows == 0) {
                    print "NO MATCHES FOR $a.\n\n";
                }
            }
        }

```

```

else {
    while ($pmid, $meta = $sthpmid->fetchrow_array()){
        $e= "$pmid|$meta";
        push(@ea, $e);
    }
    foreach $u(@ea){
        if ($u =~ /(.)\|(.)/){
            $pm=$1;
            $me=$2;
        }

    }

    #print "$pm, $i, $a, $me\n";

    if ( !$sthambiente->execute($pm, $i, $a, $me)) {
        warn "error al insertar: " . $DBI::errstr;
        $errflag = 1;
        exit;
    }
    if ( !$errflag ) {
        $dbh->commit();
    }
    else {
        $dbh->rollback();
    }
}
}
}

```

Fin del bucle de cada archivo

#Cerrar todas las transacciones con las tablas

open(MUES, '+>id_ambiente.txt'); #archivo para guardar el ultimo numero de muestra

if (!open(MUES, 'id_ambiente.txt')) {die "Error al abrir el fichero 'id_ambiente.txt'";}

print MUES "\$i\n";

close(MUES);

\$sthambiente->finish() unless (\$DBI::err);


```
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);
```

```
$sthpmid->finish() unless ($DBI::err);
```

```
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);
```

```
#Cerrar la coneccion con la base de datos y el archivo
```

```
$dbh->disconnect() || warn " Fallo al desconectar . Error : $DBI::errstr \n ";
```

```
close(UNIQ);
```

```
exit;
```

-acconteofasta.pl:

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

```
use DBI;
```

```
use strict;
```

```
use warnings;
```

```
#cuando se actualiza la base de datos
```

```
open( FAST, 'fastaact' );
```

```
if ( !open( FAST, 'fastaact' ) ) {die "Error al abrir el fichero 'fastaact'";}
```

```
my $numfa = 0;
```

```
my $i = 0;
```

```
foreach (<FAST>) {
```

```
    open (PART, ">fas$i.txt")||die "ERROR: no se puede leer o crear el  
archivo titulo\n";
```

```
    my $line = $_;
```

```
    chomp($line);
```

```
    if ($line =~ /^>/){
```

```
        $numfa++;
```

```
    }
```

```
    print PART $line;
```

```
    if ($numfa == 50000){
```

```
        if ($line !~ /.+/){
```

```
            close(PART);
```

```
            $i++;
```

```
        }
```

```
    }
```

```
}
```

```
close(FAST);  
close(PART);
```

```
exit;
```

-bdmicro.sql:

```
CREATE DATABASE microdb;
```

```
CREATE TABLE micro (  
    Titulo TEXT ,  
    Autores TEXT ,  
    Pmid VARCHAR(12) NOT NULL,  
    Fecha VARCHAR(50) NOT NULL,  
    Isolation_source TEXT ,  
    Metadatos TEXT,  
    Gen VARCHAR(100) NOT NULL ,  
    Secuencia TEXT NOT NULL ,  
    UNIQUE(Pmid),  
    PRIMARY KEY (Pmid)  
);
```

```
CREATE TABLE ambiente (  
    Pmid VARCHAR(100) NOT NULL,  
    ID_ambiente VARCHAR(100),  
    Nombre_tipo TEXT ,  
    Metadata TEXT ,  
    FOREIGN KEY (Pmid) REFERENCES micro(Pmid) ON DELETE CASCADE  
);
```

```
CREATE TABLE muestra (  
    Pmid VARCHAR(100) NOT NULL,  
    Titulo TEXT ,  
    Autores TEXT ,  
    NUM_seq VARCHAR(10),  
    ID_muestra VARCHAR(100),  
    FOREIGN KEY (Pmid) REFERENCES micro(Pmid) ON DELETE CASCADE  
);
```

```

CREATE TABLE especies (
    ID_cluster VARCHAR(100),
    Taxon TEXT ,
    NUM_seq VARCHAR(10),
    ID_seq_representante VARCHAR(100),
    PRIMARY KEY (ID_seq_representante),
    FOREIGN KEY (ID_seq_representante) REFERENCES secuencias(Pmid)
ON DELETE CASCADE
);

```

```

CREATE TABLE Taxones (
    ID_seq_representante VARCHAR(100) NOT NULL,
    superkingdom_besthit TEXT ,
    superkingdom_bestAVG TEXT ,
    superkingdom_ID VARCHAR(8) ,
    phylum_besthit TEXT ,
    phylum_bestAVG TEXT ,
    phylum_ID VARCHAR(8) ,
    class_besthit TEXT ,
    class_bestAVG TEXT ,
    class_ID VARCHAR(8) ,
    orderbesthit TEXT ,
    orderbestAVG TEXT ,
    orderID VARCHAR(8) ,
    family_besthit TEXT ,
    family_bestAVG TEXT ,
    family_ID VARCHAR(8) ,
    genus_besthit TEXT ,
    genus_bestAVG TEXT ,
    genus_ID VARCHAR(8) ,
    species_besthit TEXT ,
    species_bestAVG TEXT ,
    species_ID VARCHAR(8) ,
    FOREIGN KEY (ID_seq_representante) REFERENCES
especies(ID_seq_representante) ON DELETE CASCADE
);

```

```

CREATE TABLE secuencias (

```

```

ID_cluster VARCHAR(100) NOT NULL,
Pmid VARCHAR(100) NOT NULL,
longitud_seq VARCHAR(10),
secuencia TEXT ,
FOREIGN KEY (Pmid) REFERENCES micro(Pmid) ON DELETE CASCADE
);

```

+ANEXO III (Programas para el análisis de la base de datos):

-esp_seq.pl:

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

```

use DBI;
use strict;
use warnings;

```

```
$|=1;
```

```

my $dbh = DBI->connect( "dbi:Pg:dbname=microdb;host=localhost","agirre",
"", { RaiseError => 0, AutoCommit => 0 } );

```

```

unless ( defined($dbh) ) {die "Ha habido un problema al conectar con la base
de datos:" . $DBI::errstr unless ( defined($dbh) );}

```

```
my $sthfech = $dbh->prepare("SELECT Fecha FROM micro" );
```

```

unless ( defined ( $sthfech) ) {die "no se pueden prepara las datos para
insertarlos en las tablas\n";}

```

```

open( FECHA, '+>FECHA.txt' );      #archivo para guardar el ultimo numero
de muestra

```

```

if ( !open( FECHA, 'FECHA.txt' ) ) {die "Error al abrir el fichero
'FECHA.txt'";}

```

```

my $e="";
my @pmid=();
my $seq="";
my $i="";
my @clus=();
my @mues=();

```

```

my $clus="";
my $mues="";
my $u="";
my $o="";
my $fech="";
my @fecha=();
my $ano="";
my $ano1="";

```

```

unless ( $sthfech->execute) {die "Se ha producido un problema al conectar
con la base de datos: " . $DBI::errstr unless ( defined($dbh) );}

```

```

if ($sthfech->rows == 0) {
    print "mal\n";
}
else {
    while (my ($fech) = $sthfech->fetchrow_array()) {
        push(@fecha, $fech);
    }
}
foreach $u(@fecha){
    if ($u=~/\d{2}\-\w{3}\-(\d{4})/){
        $ano=$1;
    }
    print FECHA "$ano\n";
}

```

```

#### Cortar la comunicacion con las tablas

```

```

$sthfech->finish() unless ($DBI::err);
warn "Error de consulta: " . $DBI::errstr if ($DBI::err);

```

```

## Desconectar base de datos

```

```

$dbh->disconnect || warn " Fallo al desconectar . Error : $DBI::errstr \n ";

```

```

close(FECHA);

```

```

system ('sort FECHA.txt > sortFECHA.txt');

```

```
system ('uniq sortFECHA > uniqFECHA.txt');
```

```
exit;
```

-conteofecha.pl:

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

```
use strict;
```

```
use warnings;
```

```
$|=1;
```

```
open( FECHA, 'FECHA.txt' );
```

```
    if ( !open( FECHA, 'FECHA.txt' ) ) {die "Error al abrir el fichero  
'FECHA.txt'";}
```

```
    my @fecha = <FECHA>;
```

```
    chomp @fecha;
```

```
open( UNIQ, 'uniqFECHA.txt' );
```

```
    if ( !open( UNIQ, 'uniqFECHA.txt' ) ) {die "Error al abrir el fichero  
'uniqFECHA.txt'";}
```

```
    my @uni = <UNIQ>;
```

```
    chomp @uni;
```

```
#crear el archivo de los titulos y autores sin repeticiones
```

```
open (CONT, '>SEQfecha.txt');
```

```
    if ( !open( CONT, 'SEQfecha.txt' ) ) {die "Error al abrir el fichero  
'SEQfecha.txt'";}
```

```
my $a="";
```

```
my $e="";
```

```
my @ano=();
```

```
my $num="";
```

```
my $tot=0;
```

```
foreach $a(@uni){
```

```
    foreach $e (@fecha){
```

```
        if ($a eq $e){
```

```
            push (@ano, $e);
```

```
        }
```

```
    }
```

```

$num = scalar@ano;
$tot= $tot+$num;
print CONT "$a    $num $tot\n";
@ano=();
}

```

```

#close(CONT);
close(FECHA);
close(UNIQ);
close(CONT)
exit;

```

-fecha_seq.R:

```

tab <- read.table(file = "SEQfecha.txt",
                  header = FALSE, sep = "\t")

```

```

colnames(tab) = c("FECHA", "SECUENCIAS", "SECUENCIAS TOTALES")

```

```

tab
fecha <- tab[,1]
secuencias <- tab[,2]
secuencias_total <- tab[,3]
FECHAS <- fecha[fecha<=2011]
SECUENCIAS <- secuencias[fecha<=2011]
SEQ_TOTAL <- secuencias_total[fecha<=2011]

```

```

plot(FECHAS, SEQ_TOTAL)
lines(FECHAS, SEQ_TOTAL, col=3)
lines(FECHAS, SECUENCIAS, col=6)

```