HERRAMIENTA BIOINFORMÁTICA BAJO PERL Y LINUX, PARA LA DETERMINACIÓN DE PATRONES DE ADN, IMPLICADOS EN LA REGULACIÓN TRANSCRIPCIONAL DE LOS GENES

Carlos Andrés Pérez G.*

Evaluadores: **Diego Fernando Marin****
Raúl Alberto Cuervo Mulet***

Tipo de Artículo: **Investigación Científica y Tecnológica**

RESUMEN

El presente artículo expone un programa elaborado en PERL y ejecutado bajo Linux, que permite extraer secuencias flanqueadoras de genes de genomas completos de procariotas y compararlas entre sí, con el objetivo de encontrar posibles secuencias comunes de unión a factores de transcripción (proteínas que regulan la transcripción luego de una translocación nuclear debido a una interacción específica con ADN o por una interacción estequiométrica con una proteína que puede unirse al complejo secuencia- ADN específico-proteína[1]). Los conjuntos de genes se han organizado a partir de experimentos de micro arreglos de ADN.

PALABRAS CLAVE

Bioinformática, ADN, PERL, transcripción, Factores de transcripción.

---

* Biólogo genetista, especialista en Simulación Molecular; maestría en Bioinformática y estudios doctorales en Biotecnología; Director del Centro de Investigaciones en Ciencias Básicas, Ambientales y Desarrollo Tecnológico (CICBA) y del Grupo de Investigación en Biotecnología y Medio Ambiente (GIBMA), Universidad Santiago de Cali. Profesor de dedicación exclusiva de la Universidad Santiago de Cali. caperez@usc.edu.co

** Ingeniero de Sistemas Universidad San Buenaventura Cali (1992), Especialista en Procesos para el Desarrollo de Software Universidad San Buenaventura Cali (2003), Estudiante del Master en Ingeniería de Software en la Universidad Politécnica de Madrid. Vinculado a los grupos de investigación en Computación Móvil y Tecnología de Banda Ancha –COMBA– y Desarrollo de Software y Software Libre –GIDESOFT–, Coordinador de los Semilleros de Seguridad Informática, y Software Libre. Profesor de Tiempo Completo Especial de la Universidad Santiago de Cali. diego.marin01@usc.edu.co

*** Biólogo Universidad del Valle (1998), Mejor trabajo de investigación en biología molecular para estudiante de pregrado, Asociación Colombiana de Ciencias Biológicas - A.C.C.B. -. Maestría en Ciencias Biológicas de la Universidad del Valle (2002). Tesis de Maestría Meritoria, Universidad Del Valle. Profesor tiempo completo Universidad San Buenaventura Cali. racuervo@usbcali.edu.co

[1] Wingender,E. (1997). Mol. Biol. 31, 483-497.

ABSTRACT

This article showcases a PERL program and running under linux, which allows the extraction of the flanking sequences of complete prokaryotic genome genes, and compare them to each other to find possible union sequences of union to transcription factors (proteins which control the transcription after a nuclear translocation due to a specific DNA interaction or by a stochiometric interaction with a protein capable of attaching to the complex sequence – specific DNA – protein). The gene sets have been organized from DNA micro arrays experiments.

Key Words: Bioinformatics, ADN, PERL, Transcription, Transcription Factors.

INTRODUCCION

El ADN (ácido desoxirribonucleico) es la molécula que contiene la información hereditaria sobre la codificación de las proteínas, las cuales no se producen en todo momento, sino, cuando los organismos lo requieren. Esta capacidad regulatoria la contiene el mismo ADN, mediante secuencias cortas a las cuales se unen proteínas específicas denominadas factores de transcripción que promueve o bloquean la producción proteica. Localizar estas secuencias cortas por métodos experimentales es bastante dificultoso, por tanto, se ha elaborado un programa en PERL y ejecutado bajo Linux, que permite extraer secuencias flanqueadoras de genes de genomas completos (dotación completa de genes de un organismo) de bacterias y compararlas entre sí, con el objetivo de encontrar posibles secuencias comunes de unión a factores de transcripción y ayudar a entender como determinados genes son regulados. La localización de estas secuencias reguladoras y su alteración genética mediante mutaciones dirigidas, permitiría bloquear o aumentar la producción de proteínas especificas de interés biotecnológico en el área de la salud o comercial. Es importante resaltar que el ser humano utiliza estos microorganismos en actividades que van desde

la producción de quesos y modificación del sabor de algunos alimentos hasta la generación de insulina humana para el tratamiento de la diabetes.

Adicionalmente, la sistematización de los procesos de secuenciación, sumado a la cantidad de proyectos en esta área y a la facilidad de acceder a esta tecnología, ha aumentado de manera vertiginosa el número de genomas completos secuenciados de microorganismos, sin embargo, este hecho contrasta con el análisis a estos resultados; se conoce la mayor parte de la secuencia del genoma, pero no, cuales regiones específicas controlan los procesos relacionados con la síntesis de proteínas.

Los organismos poseen genomas (sumatoria de todos los genes) que están conformados por ADN. Pero este tipo de estructuras son mucho más que adenina, guanina, citosina y timina. El material genético contiene genes, cuya definición ha ido evolucionando conforme vamos entendiendo más sobre la dinámica de la información genética a lo largo del tiempo, tanto en el plano evolutivo como hereditario.

El gen ha pasado de ser una simple unidad hereditaria a ser el segmento de ADN que produce una transcripción funcional (ARN ya sea mensajero, ribosómico, nuclear o de transferencia) en un momento y lugar determinados.

La consideración de las variables espacio temporales, hace preguntarnos cuales son los elementos que participan en la expresión genética? y cómo es el proceso?, indicándonos que existen factores reguladores de la transcripción y traducción de la información biológica.

Por tanto, la generación de ARN que codificará proteínas, estará sujeta a diferentes tipos de secuencias de ADN; diferentes a las exónicas (secuencias codíficantes de proteínas), entre las que se pueden destacar la región promotora a la cual se liga la enzima ARN polimerasa, encargada de sintetizar el ARN mensajero e indicando el sitio de inicio de la transcripción.

En procariotas, además de la región promotora, está la secuencia operador que se encuentra cerca al promotor, cuya función es la de ser un sitio de unión de proteínas reguladoras represoras o activadoras del operón, conformado por genes involucrados en un mismo proceso bioquímico en una misma unidad de transcripción, por tanto, la expresión de todos los genes se regula de manera coordinada. Por tanto, la unidad genética de expresión coordinada se denomina operón.

1. PERL Y LA BIOLOGIA MOLECULAR

"PERL (Practical Extraction and Report Language), es un lenguaje estructurado que tiene la posibilidad de manejar datos de tipo escalar, lista o arreglos, hashes y ficheros. Así mismo es muy versátil en su aplicabilidad debido a la facilidad que tiene de generar funciones que pueden ser evaluadas en cualquier tipo de datos, adicional a su capacidad de integrarse a sistemas de bases de datos, páginas Web dinámicas y manejo de sistemas operativos como Linux permitiendo asistir al usuario con tareas comunes muy pesadas para el Shell y muy complicadas para codificarlas en algún lenguaje tipo UNIX"[2].

Larry Wall comenzó el desarrolló de Perl en 1986. Perl es un intérprete de código, un lenguaje de programación, pensado inicialmente para recoger en un único lenguaje ampliado las características de varios, Perl sobrepasó con creces sus objetivos iniciales especialmente por el impulso que recibió su inmediato uso como respondedor en páginas Web. Antes que Perl se utilizaban awk, sed y grep para analizar ficheros y extraer de ellos información a la vez que según la información extraída se ejecutaban acciones. Perl reunió las posibilidades de estas ideas UNIX en un sólo programa ampliando y modernizando cada función y haciéndolo realmente mucho más potentes, prácticos, más rápidos y finalmente ha llegado a ser utilizado en muchos más ámbitos de los que ni siquiera imaginara su autor originalmente.

[2] Simulación computacional de secuencias de ADN bajo Linux y Perl
http://www.linuxfocus.org/Castellano/August2005/article388.shtml

PERL es un lenguaje de programación gratuito y se puede ejecutar en cualquiera de los sistemas operativos que generalmente se encuentran en los laboratorios de investigaciones biológicas, presentando una gran integralidad con Unix y Linux. Gran parte de la información biológica que va desde la estructura de proteínas en formato PDB, hasta la secuencia y características de genomas completos, se organizan en archivos de textos planos lo que hace de PERL una poderosa herramienta de extracción de información, que en la mayoría de los casos, no es editada y ha sido escrita directamente por los equipos experimentales, por ejemplo, los de secuenciación, que diariamente arrojan la codificación de miles de secuencias de ADN, pero que no se conoce su significado biológico.

Pero PERL vas más allá de extracción de información y su organización a partir archivos con información biológica. Es un lenguaje de alto nivel, que permite complementar y generar nuevas funciones analíticas mediante módulos, integrando extracción con análisis de información en un solo lenguaje. Esto ha conducido que varios de los algoritmos más importantes en Bioinformática, como es el BLAST (Basic Local Alignment Search Tool, http://www.ncbi.nlm.nih.gov/blast/Blast.cgi), estén escritos en PERL y existan una gran variedad de módulos para Bioinformática bajo el proyecto BIOPERL (http://www.bioperl.org/wiki/Main_Page).

A nivel de investigación, Perl tiene un gran interés como herramienta analítica debido a que se pueden programar estructuras de datos muy complejas y existen librerías que abarcan prácticamente todos los aspectos de la informática actual (http://www.xahlee.org/PerlMathematica_dir/perlMathematica.html). Sus módulos se pueden integrar con librerías de C, GTK (gráficos), lo cual aumenta el rango de programas integrables en un solo lenguaje.

2. DETERMINACION DE PATRONES COMUNES DE ADN

El siguiente programa en lenguaje Perl, se desarrolló con la colaboración de los Doctores Juan Falgueras Cano y Antonio J. Pérez Pulido de la Universidad de Málaga – España.

Para ejecutarse el programa, se necesita que se introduzcan los nombres de los archivos que contienen el genoma del organismo a estudiar, el listado de genes de los cuales se tiene alguna evidencia de expresarse en condiciones ambientales similares, la longitud mínima de las secuencias que debe ser idéntica y el tamaño de las secuencias flanqueadoras de genes que se quiere comparar.

Previamente a la ejecución de l programa, se deben crear las carpetas 5´ y 3´, en las que se guardarán las secuencias flanqueadoras correspondientes para cada gen. El programa se apoya en el software lalign35 (http://fasta.bioch.virginia.edu/fasta_www2/fasta_down.shtml), el cual es ejecutado automáticamente, con el objeto de comparar cada una de las secuencias entre si de cada carpeta.

Para instalar los programas fasta bajo Linux, se descomprimen mediante la siguiente línea de comando:

```
gunzip -c fasta3.tar.gz | tar xvf –
```

Para identificar el “Makefile” de la maquina Linux, se utiliza:

```
make -f Makefile.linux fasta35
```

Para compilar todos los programas:

```
make -f Makefile.linux_sse2 all
```

Los datos resultantes de las comparaciones se guardan en los archivos **ResultadosLalign3´.txt** y **ResultadosLalign5´.txt**. Posteriormente, el programa filtra los resultados a partir de los valores suministrados por el usuario al inicio de la ejecución

del programa, guardando las secuencias seleccionadas en los archivos **ResultadosComparacion3´.txt** y **ResultadosComparacion5´.txt.**

Componentes del programa:

Cabecera del programa:

```
#!/usr/bin/env perl –w
# Programa para determinar secuencias reguladoras
# de la transcripción - alineamiento.pl
```

Funciones:

El programa busca secuencias flanqueadoras de genes de interés, las compara y determina patrones comunes. Solicita el nombre del archivo con el genoma del organismo en estudio, y el listado de genes.

El archivo con el genoma es leído línea a línea y toma sobre la marcha la cabecera de los genes de interés, para cada uno se recoge:

```
#     - nombre
#     - si es complementario (0 ó 1)
#     - comienzo
#     - fin
```

Posteriormente, el ADN es recogido en un @rray de líneas y luego se lleva a una cadena. El programa ejecuta el align, recogiendo la salida en una variable que posteriormente se examina. Se determina el porcentaje de identidad y presenta los resultados que sobrepasen el umbral establecido. Además sobre la marcha toma el comienzo y el fin de la coincidencia de las cadenas.

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

Variables Globales:

Longitud de las secuencias 3' y 5' a extraer:

```
my $LONGSEC = 100;  # por defecto
```

Porcentaje mínimo de coincidencia:

```
my $PORCENMIN = 75;  # por defecto
```

Longitud mínima del alineamiento:

```
my $LONGMINCOIN = 7;  # por defecto
```

Nombres de los ficheros de ADN y de los genes:

```
# Cadenas vacías.
my $dnafich = "";
my $genfich = "";
```

Análisis de los argumentos de comando:

Se llaman de la siguiente manera:

```
[longacierto]
if (@ARGV == 0) {
     my $ent;
     print("Fichero con dna? :");
     chomp($dnafich = <STDIN>);
      &mensajeComoUsarYSalir() if ($dnafich  eq "");
     print("Fichero con genes a buscar? :");
     chomp($genfich = <STDIN>);
      &mensajeComoUsarYSalir() if ($genfich  eq "");
     print("Longitudes de las secuencias de trabajo? [$LONGSEC] :");
     chomp($ent = <STDIN>);
     $LONGSEC = $ent if ($ent ne "");
     print("Porcentaje minimo de coincidencia? [$PORCENMIN] :");
     chomp($ent = <STDIN>);
     $PORCENMIN = $ent if ($ent ne "");
     print("Longitud minima de coincidencia? [$LONGMINCOIN] :");
     chomp($ent = <STDIN>);
     $LONGMINCOIN = $ent if ($ent ne "");
} else {
     if (@ARGV < 2) {
         mensajeComoUsarYSalir();
     }
     $dnafich = $ARGV[0];
     $genfich = $ARGV[1];
     if ($ARGV[2]) {
         $LONGSEC = $ARGV[2];
         if ($ARGV[3]) {
            $PORCENMIN = $ARGV[3];
            if ($ARGV[4]) {
               $LONGMINCOIN = $ARGV[4];
            }
         }
```

```
     }
}

print("$dnafich $genfich $LONGSEC $PORCENMIN $LONGMINCOIN\n");
```

Lectura de ficheros:

Lectura directa de todo el fichero de genes. El listado se lleva a array:

```
open(GENF, $genfich) or die("ERROR: <$genfich> no abre\n");
chomp(my @genesnomb = <GENF>);
close(GENF);
```

Análisis del fichero con el genoma:

```
open(DNAF, $dnafich) or die("ERROR: <$dnafich> no abre\n");
my($escomplemento, $pos0, $pos1, $engen, %gen);
```

Línea a línea la cabecera se extraen todos los genes:

```
$engen = 0;
while (<DNAF>) {
```

Carácter numérico, aparece una o más veces:

```
   if (/^FT   CDS +(complement)?\(?(\d+)\)?\.\.\(?(\d+)\)?/) {
      $escomplemento = ($1)?1:0;
      $pos0 = $2;
      $pos1 = $3;
      $engen = 2;      # líneas que faltan hasta nombre del gen
      next;            # análisis de esta línea
      } elsif ($engen > 0) {
           if ($engen == 2) {
               --$engen;    # una línea menos
               next;
           } else {
             # Carácter alfa numérico.
                if (/^FT                   \/gene="(\w+)/) {
                       #  aquí se pone el filtro para que
                       #  sólo recoja los genes que hay
                       #  en @genesnomb
                       $gen{$1} = [$escomplemento, $pos0, $pos1];
                }
                $engen = 0;
             }
           } elsif (/^SQ   /) {
                last;      # Se finaliza la selección de cabeceras, se
                           # continua con la obtención de las secuencias.
             }
    }
```

Se recoge todo el genoma en un arreglo:

```
my @dna = <DNAF>;
close(DNAF);
```

El arreglo se convierte en cadena:

```
my $DNA = join('', @dna);
@dna = ();
# limpiar
```

Si se pueden dar además de los nucleótidos acgt otros errores, se deben considerar y borrarlos con $DNA =~ tr/\s\d//d:

```
$DNA =~ tr/acgt//dc;
# Genes encontrados (TODOS los que había)
# Número de elementos del arreglo.
my @genesTODOS = keys(%gen);
printf("Localizando %d de un total de %d genes\n", $#genesnomb+1,
$#genesTODOS+1);
```

Proceso de la secuencia:

```
# Secuencia, nombre del gen de interés, si es o no complementario
# (0 o 1), Posición inicial, posición final.

foreach my $g (@genesnomb) {
     my ($tres, $cinco) = &secuencias(\$DNA, $g, $gen{$g}[0], $gen{$g}
     [1], $gen{$g}[2]);
     # Fichero 3' en el directorio 3s/
     open(OUT, ">3s/$g") or die("ERROR: 3s/$g NO abre\n");
     # cabecera
     printf(OUT ">%s\n",$g);
     # secuencia 3'
     print(OUT $tres);
     close(OUT);

     # Fichero 5' en el directorio 5s/
     open(OUT, ">5s/$g") or die("ERROR: 5s/$g NO abre\n");
     # cabecera
     printf(OUT ">%s\n",$g);
     # secuencia 5'
     print(OUT $cinco);
     close(OUT);
     # Mostrar estadísticas en la pantalla.
```

Líneas para mostrar estadísticas en la pantalla:

```
   printf("\t%5s de %7d a %7d (%s complementario)\n",
          $g, $gen{$g}[1], $gen{$g}[2], ($gen{$g}[0])?"si":"no");
}
```

Ejecución del Lalign

Definición de variables.

```
my($i, $j);
my($primera, $segunda);
my($sal);
my($porcen);
my($loncon);
my($linea1);
for ($i=0; $i <= $#genesnomb; $i++) {
  for ($j=$i+1; $j <= $#genesnomb; $j++) {
      $primera = $genesnomb[$i];
      $segunda = $genesnomb[$j];
      $sal = `lalign  3s/$primera 3s/$segunda`;
      open(OUTFILE, "+>>ResultadosLalign3´.txt") or die ("No se puede
      abrir archivo  de resultados : $!");
      print(OUTFILE  "-----------------------------------------------------
      -----\n");
      print(OUTFILE  "-----------------------------------------------------
      -----\n");
      print(OUTFILE  "Comparacion entre el gen $primera y el gen
      $segunda\n\n");
      print(OUTFILE  "$sal\n\n");
      close(OUTFILE) or die ("No se puede cerrar el archivo de salida
      output: $!");
   }
}

open(LAL, "ResultadosLalign3´.txt") or die("ERROR:
  <ResultadosLalign3´.txt> no abre\n");

while (<LAL>) {
   if (/\s+([\d\.]+)\% .* (\d+) nt overlap;/) {
      $porcen = $1;
      $loncon = $2;
      if ($porcen >= $PORCENMIN && $loncon >= $LONGMINCOIN){
         open(OUTFILE2, "+>>ResultadosComparacion3´.txt") or die ("No se
              puede abrir archivo  de resultados : $!");
         print(OUTFILE2 "El porcentaje de identidad es $porcen % y el
               numero de NT alineados es $loncon\n");
         print(OUTFILE2 "-----------------------------------------------
               ----------------------------\n");
         $linea1 = <LAL>;
         print (OUTFILE2  "$linea1");
         $linea1 = <LAL>;
         print (OUTFILE2  "$linea1");
         $linea1 = <LAL>;
         print (OUTFILE2  ">$linea1");
         $linea1 = <LAL>;
         print (OUTFILE2  " $linea1");
         $linea1 = <LAL>;
         print (OUTFILE2  ">$linea1");
         $linea1 = <LAL>;
```

```
        print (OUTFILE2  "$linea1");
        $linea1 = <LAL>;
        print (OUTFILE2  "$linea1\n\n");
        $linea1 = <LAL>;

        close(OUTFILE2) or die ("No se puede cerrar el archivo de salida
        output: $!");
    }
   }
  }
  ###################################################################

my($y, $z);
my($primera2, $segunda2);
my($sal2);
my($porcen2);
my($loncon2);
my($linea12);
for ($y=0; $y <= $#genesnomb; $y++) {
    for ($z=$y+1; $z <= $#genesnomb; $z++) {
        $primera2 = $genesnomb[$y];
        $segunda2 = $genesnomb[$z];
        $sal2 = `lalign  5s/$primera2 5s/$segunda2`;
        open(OUTFILE, "+>>ResultadosLalign5´.txt") or die ("No se puede
        abrir archivo  de resultados : $!");
        print(OUTFILE  "-------------------------------------------------
        --------\n");
        print(OUTFILE  "-------------------------------------------------
        --------\n");
        print(OUTFILE  "Comparacion entre el gen $primera2 y el gen
        $segunda2\n\n");
        print(OUTFILE  "$sal2\n\n");
        close(OUTFILE) or die ("No se puede cerrar el archivo de salida
        output: $!");
     }
  }

open(LAL, "ResultadosLalign5´.txt") or die("ERROR:
<ResultadosLalign5´.txt> no abre\n");

while (<LAL>) {
     if (/\s+([\d\.]+)\% .* (\d+) nt overlap;/) {
          $porcen2 = $1;
          $loncon2 = $2;
          if ($porcen2 >= $PORCENMIN && $loncon2 >= $LONGMINCOIN){
               open(OUTFILE2, "+>>ResultadosComparacion5´.txt") or die
               ("No se puede abrir archivo  de resultados : $!");
               print(OUTFILE2 "El porcentaje de identidad es $porcen2 %
               y el numero de NT alineados es $loncon2\n");
               print(OUTFILE2 "-----------------------------------------
               ------------------------------------\n");
               $linea12 = <LAL>;
               print (OUTFILE2  "$linea12");
               $linea12 = <LAL>;
```

```
                print (OUTFILE2  "$linea12");
                $linea12 = <LAL>;
                print (OUTFILE2  ">$linea12");
                $linea12 = <LAL>;
                print (OUTFILE2  " $linea12");
                $linea12 = <LAL>;
                print (OUTFILE2  ">$linea12");
                $linea12 = <LAL>;
                print (OUTFILE2  "$linea12");
                $linea12 = <LAL>;
                print (OUTFILE2  "$linea12\n\n");
                $linea12 = <LAL>;
                close(OUTFILE2) or die ("No se puede cerrar el archivo de
                salida  output: $!");
           }
     }
}
```

Determinación de patrones Comunes:

```
my($nomp3);
my($seqXp3);
my(@lineasunidasp3);
my($a, $b,$q);
my($nom1,$nom2);
my($primeraseqp3);
my($segundaseqp3);
@lineasunidasp3 = ();
open(REC, "ResultadosComparacion3´.txt") or die("ERROR:
<ResultadosComparacion3´.txt> no abre\n");
while (<REC>) {
     if (/^>(.+)   (.+)/) {
          $nomp3 =$1;
          $seqXp3 =$2;
          # En el arreglo @lineasunidasp3, se coloca el nombre del gen
          # segido de la secuencia.
          push (@lineasunidasp3,$nomp3);
          push (@lineasunidasp3,$seqXp3);
     }
}
close(REC);

open(OUTFILE3, "+>>alineamientosComunes3´.txt") or die ("No se puede
abrir archivo  de alineamientos Comunes : $!");

$q = 0;

for ($a=1; $a <= $#lineasunidasp3; $a += 2) {
     for ($b= $a + 2; $b <= $#lineasunidasp3; $b += 2) {
          $nom1 = $lineasunidasp3[$a-1];
          $nom2 = $lineasunidasp3[$b-1];
          $primeraseqp3 = $lineasunidasp3[$a];
          $segundaseqp3 = $lineasunidasp3[$b];
          if ($primeraseqp3 =~ /$segundaseqp3/g && $nom1 ne $nom2) {
```

```
                if ($q == 0) {
                     print (OUTFILE3">$nom1  $primeraseqp3\n");
                     print (OUTFILE3">$nom2  $segundaseqp3\n");
                     $q = 2;
                }
                else {
                     print (OUTFILE3">$nom2  $segundaseqp3\n");
                }
           }
           elsif ($b >= $#lineasunidasp3 - 1 && $q == 2) {
                $q = 0;
                print (OUTFILE3"\n\n");
           }
     }
}

close(OUTFILE3) or die ("No se puede cerrar el archivo de salida  output:
$!");

#################################################################

# Patrones de las secuencias
# Flanqueadoras 5´.
my($nomp5);
my($seqXp5);
my(@lineasunidasp5);
my($c, $d,$v);
my($nom1p5,$nom2p5);
my($primeraseqp5);
my($segundaseqp5);

@lineasunidasp5 = ();

open(REC, "ResultadosComparacion5´.txt") or die("ERROR:
<ResultadosComparacion5´.txt> no abre\n");
while (<REC>) {
     if (/^>(.+)   (.+)/) {
          $nomp5 =$1;
          $seqXp5 =$2;
          push (@lineasunidasp5,$nomp5);
          push (@lineasunidasp5,$seqXp5);
     }
}
close(REC);

open(OUTFILE3, "+>>alineamientosComunes5´.txt") or die ("No se puede
abrir archivo  de alineamientos Comunes : $!");
$v = 0;
for ($c=1; $c <= $#lineasunidasp5; $c += 2) {
     for ($d= $c + 2; $d <= $#lineasunidasp5; $d += 2) {
          $nom1p5 = $lineasunidasp5[$c-1];
          $nom2p5 = $lineasunidasp5[$d-1];
          $primeraseqp5 = $lineasunidasp5[$c];
          $segundaseqp5 = $lineasunidasp5[$d];
```

```
            # Determinación de patrones Comunes.
            if ($primeraseqp5 =~ /$segundaseqp5/g && $nom1p5 ne $nom2p5) {
                  if ($v == 0) {
                        print (OUTFILE3">$nom1p5  $primeraseqp5\n");
                        print (OUTFILE3">$nom2p5  $segundaseqp5\n");
                        $v = 2;
                  }
                  else {
                        print (OUTFILE3">$nom2p5  $segundaseqp5\n");
                  }
            }
            elsif ($d >= $#lineasunidasp5 - 1 && $v == 2) {
                  $v = 0;
                  print (OUTFILE3"\n\n");
            }
      }
}

close(OUTFILE3) or die ("No se puede cerrar el archivo de salida  output:
$!");
```

Subrutinas:

Extrae y devuelve las secuencias 3' y 5' a partir de una referencia al ADN. Además de saber si es complementaria, para lo cual se llama a la rutina correspondiente que actúa sobre la cadena 3' ó 5':

```
sub secuencias {
      my($dnar, $nombre, $escomp, $orig, $fin) = @_;
      # cambio índices desde 1
      # a índices desde 0 en perl
      --$orig;
      --$fin;

      my $seq3 = substr($$dnar, $orig - $LONGSEC, $LONGSEC);
      my $seq5 = substr($$dnar, $fin+1, $LONGSEC);
      if ($escomp) {
            $seq3 = &revecomp($seq3);
            $seq5 = &revecomp($seq5);
      }
      return ($seq3, $seq5);
}
```

Obtiene y devuelve el reverso complementario de la cadena de ADN:

```
sub revecomp {
      my($seq) = $_[0];
      $seq = reverse($seq);
      $seq =~ tr/ACGTacgt/TGCAtgca/;
      return $seq;
}
```

3. CONCLUSIONES

El programa computacional descrito en el presente artículo, facilita localizar secuencias de ADN reguladoras de la transcripción, las cuales son muy difíciles de ubicar por métodos experimentales, debido a que son secuencias que no se expresan, lo que significaría someter a procesos de mutagénesis dirigida grandes longitudes de ADN, sin tener un referente sobre la región de interés e invertir mayor cantidad de tiempo y recursos económicos.

Los resultados obtenidos son fácilmente comparables, por medio de matrices de pesos de nucleótidos por posición, con las secuencias de otras especies, que por evolución natural, pueden ser altamente conservadas y realizar la misma función.

Los patrones obtenidos, permiten determinar las proteínas que se une al ADN para regular su transcripción, indicando los posibles tipos de regulación a los que se ven sometidos los genes en estudio y plantear posibles rutas reguladoras en los diferentes procesos bioquímicos vitales para los organismos.

REFERENCIAS BIBLIOGRAFICAS

1. Ben-Dor, A., Shamir, R. & Yakhini, Z. (1999) Journal of Computational Biology. 6, 281–297.
2. Brikun I, Suziedelis K, Stemmann O, Zhong R, Alikhanian L, Linkova E, Mironov A.(1996) Journal of Bacteriology. Mar;178(6):1614-22.
3. Bussemaker, H. J., Li, H. & Siggia, E. D. (2000) Proceedings of the National Academy of Sciences. USA 97,10096–10100.

4. Cai SJ, Inouye M. Journal of Biological Chemistry. 2002 Jul 5;277(27):24155-61. Epub 2002 Apr 24.
5. Eisen, M. B., Spellman, P. T., Brown, P. O. & Botstein, D. (1998) Proceedings of the National Academy of Sciences. USA 95, 14863–14868.
6. Engelhorn M, Geiselmann J. (1998) Molecular Microbiology. 30(2):443-51.
7. Felsenstein,J.(1996)Methods Enzymol. 266,418 – 427.
8. Frec.,K,Quandt,K. And Werner,T.(1997) Computer Applications In The Biosciences.13,89-87.
9. Gelfand,M.S. and Koonin,E.V.(1997) Nucleic Acids Research. 25,2430-2439.
10. Pérez, C.A., Pérez, A. & Falguera, J. (2007). Sistemas & Telemática. 10, 13 -27.
11. Wingender,E. (1997) Molecular Biology. 31, 483-497.