Soriano Rosales Eric Dilan
Márquez Zavala Elisa

Estabilidad y predicción de promotores

En este módulo de Bioinformática 2 se muestra una aplicación del modelo Nearest Neighbor para la predicción de promotores, a través de un enfoque estructural.

· PROBLEMA: Se dispone de coordenadas de una colección de marcos de lectura (ORFs), pero desconocemos la posición de sus secuencias promotoras

· SOLUCIÓN PROPUESTA: Buscar la "huella" de estabilidad molecular de los promotores en la secuencia del DNA

Para ello se empleó el algoritmo de Kanhere & Bansal (2005) que anteriormente ha demostrado ser una excelente opción para resolver este tipo de problemas.

Se tomaron como objeto de análisis secuencias del fichero K12_400_50_sites, que contiene coordenadas de ORFs de Escherichia coli con coordenadas -400,+50, con el 0 centrado cerca del codón de inicio.

Esta tarea consistió en ciertos pasos:

1) Completar el codigo fuente del programa 1.1 para implementar el predictor de Kanhere y Bansal, justificando los valores de cutoff1 y cutoff2 de acuerdo con las figuras de su artículo. Es necesario comentar el codigo explicando sus cambios, por ejemplo con http://perldoc.perl.org/perlpod.html . Pueden usar el lenguaje de programacion que quieran, siempre que haya un compilador disponible.

Los puntos de corte fueron establecidos de acuerdo a las gráficas del artículo Kanhere & Bansal (2005) que muestran los puntos óptimos con respecto a D y E1 en cuanto a sensibilidad (gráfica A) y precisión (gráfica B), D<3 y E1>18. Con el fin de obtener los valores que se encontraba entre el rango de café y amarillo, como se muestra en las siguientes imágenes.

Código

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

# Soriano Rosales Eric Dilan
# Marquez Zavala Elisa
# Nearest Neighbor dG calculator

use strict;
```

Soriano Rosales Eric Dilan
Márquez Zavala Elisa

```
# global variables
my $T          = 37; # temperature(C)
my $windowL    = 15;  # window length, http://www.biomedcentral.com/1471-2105/6/1
my @wins;  #Este vector contendra las ΔGs de cada una de las ventanas
my @promo; #Este vector contendra los posibles n que serviran de guia para encontrar los posibles promotores.
my $E1=0; #Se declara la variable que contendra los promedios de la energia libre para cada n de la primera ventana de 50nt.
my $E2=0; #Se declara la variable que contendra los promedios de la energia libre para cada n de la segunda ventana de 100nt.
my $D=0; #Se calcula la diferencia entre los dos promedios.
my $v=0;
my $izq=0;
my $der=0;
my %NNparams    = (
        # SantaLucia J (1998) PNAS 95(4): 1460-1465.
        # [NaCl] 1M, 37C & pH=7
        # H(enthalpy): kcal/mol      , S(entropy): cal/k�mol
        # stacking dinucleotides
        'AA/TT' , {'H',-7.9, 'S',-22.2},
        'AT/TA' , {'H',-7.2, 'S',-20.4},
        'TA/AT' , {'H',-7.2, 'S',-21.3},
        'CA/GT' , {'H',-8.5, 'S',-22.7},
        'GT/CA' , {'H',-8.4, 'S',-22.4},
        'CT/GA' , {'H',-7.8, 'S',-21.0},
        'GA/CT' , {'H',-8.2, 'S',-22.2},
        'CG/GC' , {'H',-10.6,'S',-27.2},
        'GC/CG' , {'H',-9.8, 'S',-24.4},
        'GG/CC' , {'H',-8.0, 'S',-19.9},
        # initiation costs
        'G'     , {'H', 0.1, 'S',-2.8 },
        'A'     , {'H', 2.3, 'S',4.1  },
        # symmetry correction
        'sym'   , {'H',   0, 'S',-1.4 } );

my $infile = $ARGV[0] || die "# usage: $0 <promoters file>\n";

print "# parameters: Temperature=$T\c Window=$windowL\n\n";

open(SEQ, $infile) || die "# cannot open input $infile : $!\n";
while(<SEQ>)
{
        if(/^(b\d{4}) \\ ([ATGC]+)/)
        {
```

Soriano Rosales Eric Dilan
Márquez Zavala Elisa

```
my ($name,$seq) = ($1,$2);
printf("----------------------------------------------\n");
printf("sequence %s (%d nts )\n",$name,length($seq));

@wins = duplex_deltaG($seq, $T, $windowL);

#print join("\n",@wins);
#print "\n";



# your code here
#Esta es la parte de calcular el D, E1 y E2.
for (my $w=0; $w<($#wins)-199; $w++)
{
	my $Ea=0; #Se declaran estas variables que son acomuladores para ir sumando las deltas de cada n. de n  a n+49
	my $Eb=0; #Con respecto de n+99 a n+199
	for(my $x=0; $x<50; $x++)
	{
		$Ea+=$wins[$w+$x];#Con se busca en el vector con las deltas de los n las posiciones de los n conrrespondientes y hacemos la sumatoria.
	}
	for(my $z=98; $z<199; $z++)
	{
		$Eb+=$wins[$w+$z];#Lo mismo pero con respecto a la otra sumatoria.
	}
	$E1=$Ea/50; #Realizamos la division de la sumatoria para sacar el promedio.
	if(-18<$E1)#Si esta dentro del punto de corte entonces entra.
	{
		$E2=$Eb/100; #Se hace el promedio con respecto a la caja de 100nt.
		$D=$E1-$E2; #Se calcula la D.
		if($D>3)#Si esta dentro del punto de corte entra.
		{
			$promo[$v]=$w; #Se guarda la posicion del n que podria ser la del promotor.
			$w+=25;  #Como pueden sobrelapar creamos una distancia de 25.
			$v++;   #Aumenta el contador para guardar en la siguiente posicion del vector promo los otros posibles.
		}
	}
```

```
		}
			printf("%s\n",$name);
			for( $v=0; $v<$#promo; $v++)#Esta parte del codigo sirve para imprimir las secuencias de los posibles promotores de longitud 15 en formato fasta para que posteriormente sea manipulable facilmente.
			{
				print "Inicio\tFinal\n";
				print $promo[$v]-400,"\t",$promo[$v]-350,"\n"; #Aqui se imprime el inicio y fin del posible promotor.
			}
		printf("-----------------------------------------------\n");



	}

}
close(SEQ);


# calculate NN free energy of a DNA duplex , dG(t) = (1000*dH - t*dS) / 1000
# parameters: 1) DNA sequence string; 2) Celsius temperature
# returns; 1) free energy scalar
# uses global hash %NNparams
sub duplex_deltaG
{
	my ($seq,$tCelsius,$windowL) = @_;

	my ($DNAstep,$nt,$dG,$total_dG) = ('','',0,0);
	my @win_seqs;#Contiene las ΔGs de la secuencia de cada una de las ventanas
	my @sequence = split(//,uc($seq));
	my $tK = 273.15 + $tCelsius;

	sub complement{ $_[0] =~ tr/ATGC/TACG/; return $_[0] }

	# add dG for overlapping dinculeotides
	for(my $i=0; $i<=($#sequence - 14); $i++)#En este bucle anidado se hace el recorrido con ventanas de 15nt
	{
		for(my $n=$i;$n<($i + 14);$n++)
		{


				my $temp=($sequence[$n].$sequence[$n+1]);#Creo una variable temoral para que no se modifique mi variable
				$DNAstep = $sequence[$n].$sequence[$n+1].'/'.
```

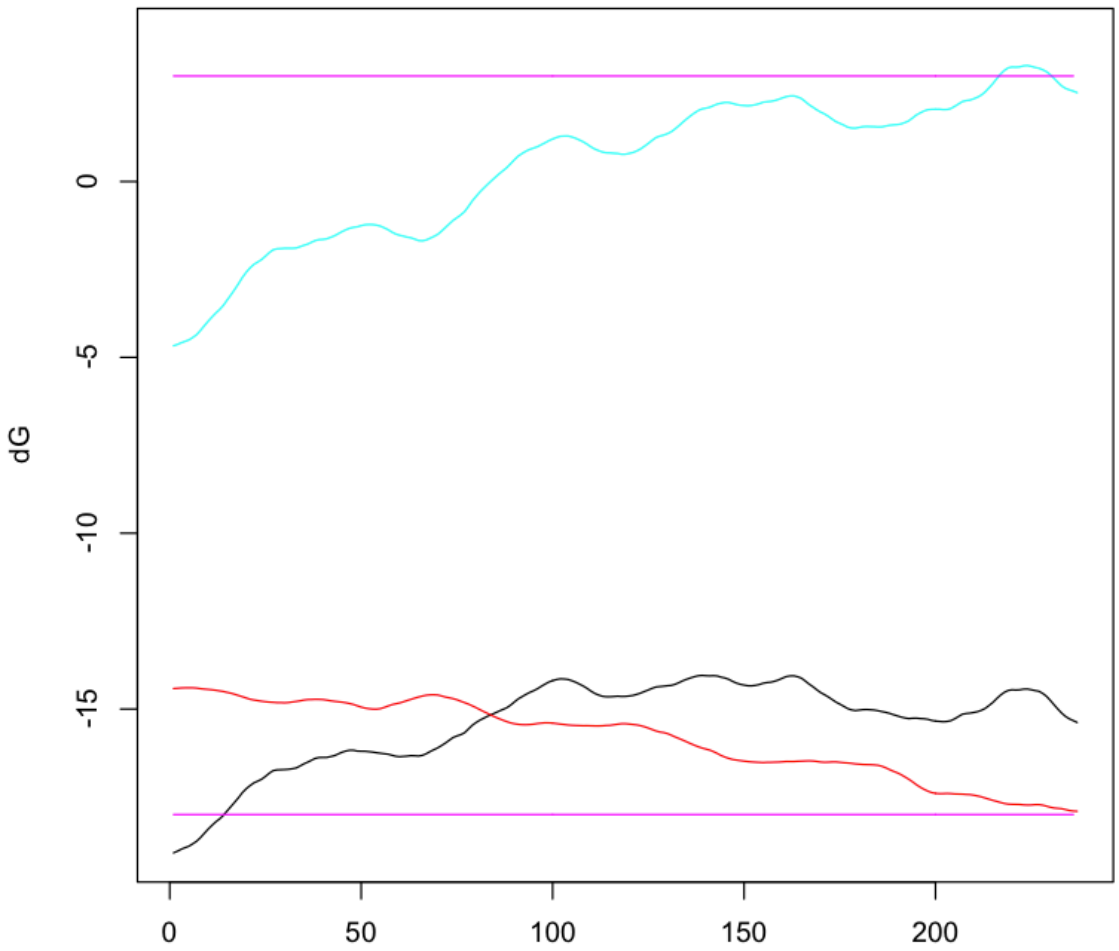
```
                        complement($temp);


                        if(!defined($NNparams{$DNAstep}))
                        {

                                $DNAstep = reverse($DNAstep);
                        }

                        $dG = ((1000*$NNparams{$DNAstep}{'H'})-
                                        ($tK*$NNparams{$DNAstep}{'S'}))
                                        / 1000 ;

                        $total_dG += $dG;
                        #printf("%s \n",$DNAstep);
                        #printf("%f \n",$dG);


                        #En esta parte agrego la corrección de la primera y última
base por ventana de 15

                        if($n==$i)
                        {
                                        $nt = $sequence[$n]; # first pair
                                        if(!defined($NNparams{$nt})){ $nt =
complement($nt) }
                                        $total_dG += ((1000*$NNparams{$nt}{'H'})-

        ($tK*$NNparams{$nt}{'S'}))
                                                                / 1000;

                                        $nt = $sequence[$n+14]; # last pair
                                        if(!defined($NNparams{$nt})){ $nt =
complement($nt) }
                                        $total_dG += ((1000*$NNparams{$nt}{'H'})-

        ($tK*$NNparams{$nt}{'S'}))
                                                                / 1000;
                                #En este bloque calculo la correccion de simetria en caso de que
haya
                                        for(my $a=$n; $a<7; $a++)
                                        {
                                                $izq .= $sequence[$n+$a];
                                                $der .= $sequence[$n+14-$a];
                                        }
```

Soriano Rosales Eric Dilan
Márquez Zavala Elisa

```
                                        $der= complement($der);

                                        if($izq eq $der)
                                        {
                                           $total_dG += ((1000*$NNparams{'sym'}{'H'})-
($tK*$NNparams{'sym'}{'S'}))/1000;
                                        }
                                    }

                }
                #printf("%f\n",$total_dG);
                $win_seqs[$i] = $total_dG;#Almaceno el total de deltas Gs de todas la
ventanas por secuencias
                $total_dG= 0;


        }

        #print join("\n",@win_seqs);
  #print"\n";
        #printf("-----------------------------------------------\n");
        return @win_seqs;
}
```

Soriano Rosales Eric Dilan
Márquez Zavala Elisa

2) Diseñar una figura donde se muestra gráficamente D, E1 y E2 para una posición n.



En esta figura, se muestra para una secuencia, cómo los valores de "D" rebasan el umbral de 3 en cierto intervalo de la secuencia. Al mismo tiempo, en este intervalo, se logra percibir que los valores de "E1" y "E2" están muy separados y no son menores al umbral de -18.

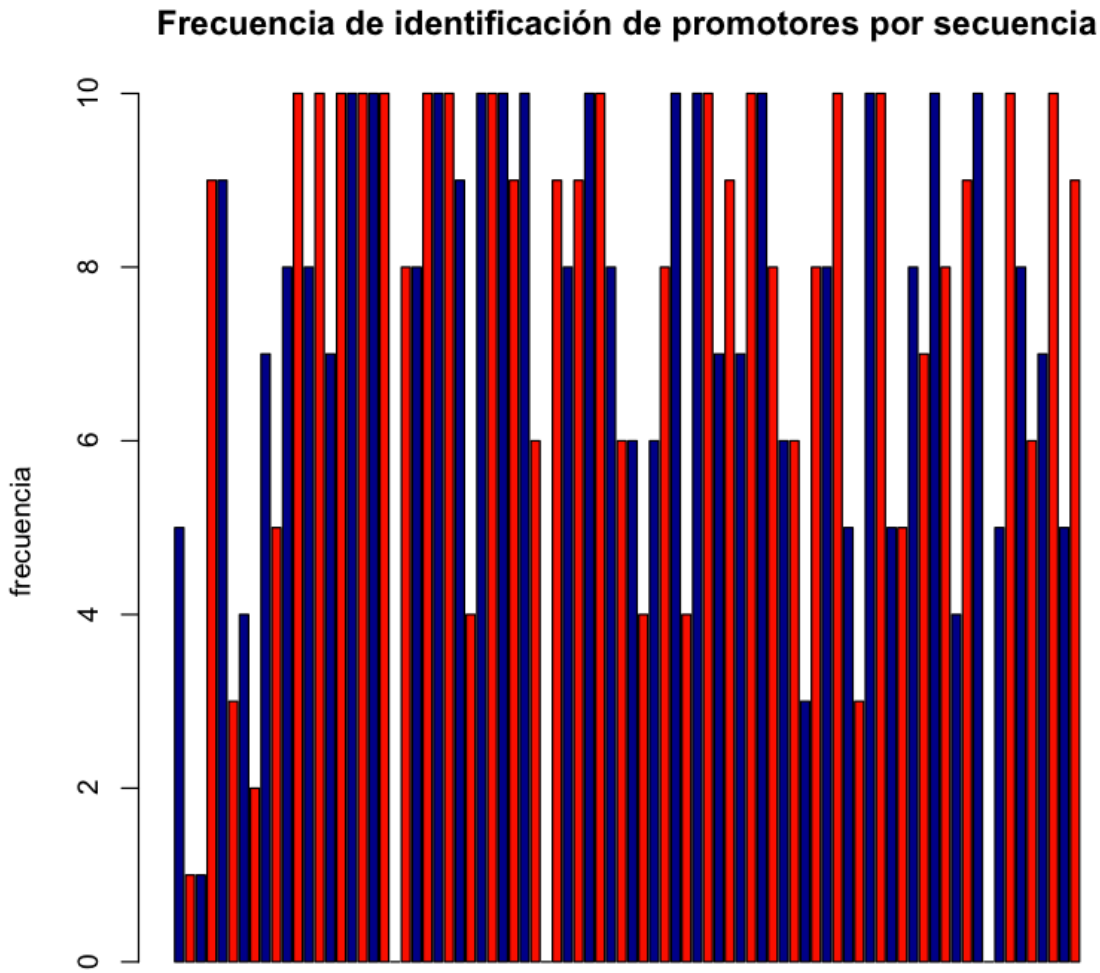
3) Predecir promotores en todas las secuencias del fichero K12_400_50_sites.

Dado que son relativamente muchas secuencias, muestro sólo los promotores predichos para la secuencia b0339:

```
sequence b0339 (451 nts )

Inicio   Final
-347     -297
Inicio   Final
-263     -213
```

Soriano Rosales Eric Dilan
Márquez Zavala Elisa

4) Graficar con que frecuencia se predicen promotores en el intervalo -400,50. Con un breve comentario de los resultados es suficiente. Se les ocurre una manera de validar sus resultados, y calcular la tasa de FPs, usando RSAT::matrix-scan?



Como se puede ver en la gráfica, el máximo de secuencia encontradas, en el intervalo indicado, es de 10. Por otro lado, el mínimo de secuencias encontradas es de 2.

Una manera de validar los resultados es de forma experimental, mutando nucleótidos que se encuentran en el intervalo, y observar si se expresa algún gene reportero en la región río abajo. También, podría reordenar mis secuencias input de manera aleatoria, ya sea por bloques o hacerlo completamente para todos los nucleótidos para observar la consistencia.
Por otro lado, para identificar falsos positivos usando el programa *matrix-quality,* podríamos usar secuencias aleatorias o secuencias biológicas que contienen ningún promotor.