Práctica 2

Anastasia Hernández y Alán Muñoz

February 15, 2016

Práctica 2

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 [http://perldoc.perl.org/perlpod.html] . Pueden usar el lenguaje de programacion que quieran, siempre que haya un compilador disponible.

Código de Perl

```
#Completar el código con el diagrama y hacer una predicción con los sitios de e.coli (formato consensus
#la transcripción)

#!/usr/bin/perl -w

# prog1.1
# Bruno Contreras-Moreira
# Nearest Neighbor dG calculator

use strict;

# global variables
my $T           = 37; # temperature(C)
my $windowL     = 15;  # window length, http://www.biomedcentral.com/1471-2105/6/1
my ($i,$j,$wind,$temp_deltas,$temp_sum,$cutoff1,$cutoff2,$prom_num,$cont);
my (@deltas,@E1,@E2,@D,@prom_pos,@prom_seq, @prom_all,@prom_all_pos);
my %NNparams    = (
    # SantaLucia J (1998) PNAS 95(4): 1460-1465.
    # [NaCl] 1M, 37C & pH=7
    # H(enthalpy): kcal/mol , S(entropy): cal/kmol
    # 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";

my $cutoff1= 3.4;
my $cutoff2= -15.99;

$cont = 0;
open(my $FILE, '>',"E1.txt");
open(my $FILE2, '>',"E2.txt");
open(my $FILE3, '>',"D.txt");



while(<SEQ>)
{
    $cont++;
    if(/^(b\d{4}) \\ ([ATGC]+)/)
    {
        my ($name,$seq) = ($1,$2);
        #Borramos los datos en nuestros vectores para hacer iteraciones.
        undef(@deltas);
        undef(@E1);
        undef(@E2);
        undef(@D);
        undef(@prom_pos);
        undef(@prom_seq);
        #hacer la ventana -> ir moviendola de n en n.
        #la n, siempre se refiere a posicion

        #Creamos las ventanas a lo largo de la secuencia, a excepción de las orillas del tamaño de mita
        for ($i=int($windowL/2); $i < length($seq)-int($windowL/2); $i++){
                $wind = substr($seq,$i-$windowL/2,$windowL);
                #llamas a la subrutina duplex_deltaG e introducimos el valor a nuestro vector de deltas
                push @deltas,duplex_deltaG($wind,$T);
        }
        #Comenzamos en nucleótido 51.
        #Hacemos el cálculo para cada n a partir de ahí hasta 150 antes de acabar
        #(para poder obtener valores E2 y D también).
        #Luego introducimos los valores E1 y E2 a los vectores @E1 y @E2

        #E1=SUM (n a n+49)*$total_dG/50
        for ($i=50; $i < length($seq)-150;$i++){
            $temp_sum = 0;
            for ($j=$i-50;$j<$i;$j++){
            $temp_sum += $deltas[$j];
            }
            $temp_deltas = $temp_sum/50;
            push(@E1,$temp_deltas);
        }
```

```
        #E2=SUM (n+99 a n+199)*$total_dG/100
        for ($i=50; $i < length($seq)-150;$i++){
            $temp_sum = 0;
            for ($j=$i+50;$j<$i+150;$j++){
            $temp_sum += $deltas[$j];
            }
            $temp_deltas = $temp_sum/100;
            push(@E2,$temp_deltas);
        }

        #Obtenemos los valores D y los introducimos a su propio vector.
        #D(n)=E1(n)-E2(n)
        $prom_num = 0;
        for ($i=0;$i < scalar(@E1);$i++){
            push(@D,$E1[$i]-$E2[$i]);

            #Despúes, usando E1 y D rescatamos los motivos de acuerdo a los valores de cutoff
            if ($D[$i]>$cutoff1 && $E1[$i]>$cutoff2) {
                #Obtenemos secuencia y valor real (sumando los windowL/2 que se omitieron al inicio)
                push @prom_seq,substr($seq,$i+50,$windowL);
                push @prom_pos,($i+50+int($windowL/2));

                $prom_num++;

                #if (n>) Si n está a 25n a -25n
                if ($prom_num>1){
                    #Si están demasiado cerca los consideramos el mismo motivo.
                    if ($prom_pos[$prom_num-1]-$prom_pos[$prom_num-2]<25){
                        pop @prom_pos;
                        pop @prom_seq;
                        $prom_num--;
                    }
                }

            }


        }




    }
    push @prom_all,@prom_seq;
    push @prom_all_pos,@prom_pos;
    if ($cont==1){
        for my $i (1..$#E1+1){
            printf $FILE "%d\t", $i;
            printf $FILE2 "%d\t", $i;
            printf $FILE3 "%d\t", $i;
        }

        printf $FILE "\n";
```

```
        printf $FILE2 "\n";
        printf $FILE3 "\n";
    }

    for my $i (0..$#E1) {
    printf $FILE "%f\t", $E1[$i];
    printf $FILE2 "%f\t", $E2[$i];
    printf $FILE3 "%f\t", $D[$i];
    }

    printf $FILE "\n";
    printf $FILE2 "\n";
    printf $FILE3 "\n";
}
close(SEQ);
close($FILE);
close($FILE2);
close($FILE3);
#print "@prom_all \n";

open(my $FILE4, '>',"predicted_promoters.txt");
printf $FILE4 "Promoter\tPos\tSequence\n";
for my $i (0..$#prom_all) {
    printf $FILE4 "%d\t%d\t%s\n",$i+1, $prom_all_pos[$i], $prom_all[$i];
}
close($FILE4);

# 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) = @_;

    my ($DNAstep,$nt,$dG,$total_dG) = ('','',0,0);
    my @sequence = split(//,uc($seq));
    my $tK = 273.15 + $tCelsius;
    my $i;

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

    # add dG for overlapping dinculeotides
    for(my $n=0;$n<$#sequence;$n++)
    {
            $DNAstep = $sequence[$n].$sequence[$n+1].'/'.
                complement($sequence[$n].$sequence[$n+1]);

            if(!defined($NNparams{$DNAstep}))
            {
                $DNAstep = reverse($DNAstep);
            }
```

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

            $total_dG += $dG;
    }

    # add correction for helix initiation
    $nt = $sequence[0]; # first pair
    if(!defined($NNparams{$nt})){ $nt = complement($nt) }
    $total_dG += ((1000*$NNparams{$nt}{'H'})-
                    ($tK*$NNparams{$nt}{'S'}))
                    / 1000;

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

    # please complete for symmetry correction
    if($seq == reverse($seq)){
        $total_dG += ((1000*$NNparams{'sym'}{'H'})-
                    ($tK*$NNparams{'sym'}{'S'}))
                    / 1000;
    }



    return $total_dG;
}

#-----------------------------Diccionario de Variables-----------------------------------------------#
=pod
$T           = 37; Temperatura
$windowL     = 15;  Tamaño de la ventana
$cutoff1= 3.4; Valor de corte para E1
$cutoff2= -15.99; Valor de corte para D
$i  Contador 1, usado para recorrer las posiciones de las que se obtendrá dG y éstos mismos valores en s
$j  Contador 2, usado para hacer sumatorias de dG en cálculos de E1 y E2
$wind Valor scalar temporal que contiene las secuencias de las ventanas para introducir a la función dup
$temp_deltas    Valor escalar temporal para guardar el valor final al obtener E1 y E2
$temp_sum   Valor escalar temporal para guardar la sumatoria de dGs y mandarlas a temp_deltas
$prom_num   Valor escalar, contador usado para evaluar posiciones mediante el vector prom_pos
$count Valor escalar, contador usado para insertar los headers a los archivos exportados hacia R
@deltas Vector unidimensional para guardar los valores dG
@E1 Vector unidimensional para guardar los valores E1
@E2 Vector unidimensional para guardar los valores E2
D   Vector unidimensional para guardar los valores D
@prom_pos Vector unidimensional para guardar las posiciones de los promotores seleccionados en cada secu
@prom_seq Vector unidimensional para guardar las secuencias de los promotores seleccionados en cada secu
@prom_all Vector unidimensional para guardar las secuencias de todos los promotores a lo largo del análi
```

```
=cut
```

= Dentro de la consideración para elegir los valores de corte (cutoff1/cutoff2), fueron considerados ambos factores: la especificidad (precisión) y la sensibilidad del método, buscando reducir los falsos positivos encontrados y, maximizando las características previamente mencionadas en los resultados. Por ende, los valores que maximizan la solución para ambos casos, serían 3 para de D (cutoff1) y -17 para E1 (cutoff2) Como puede observarse en la figura 4 del artículo; considerando además, el número de falsos positivos para minimizar más el error, se eligen los valores con frecuencia de FP de 1/16204, por lo que las cifras finales son de **3.4 para D (cutoff1) y de -15.99 para E1 (cutoff2)**. 2.76 -17.53

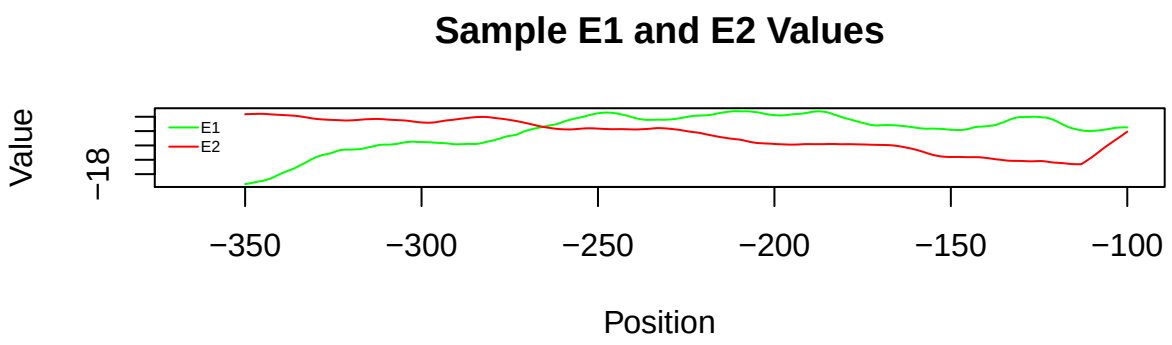
Llamamos las tablas de nuestro script de perl.

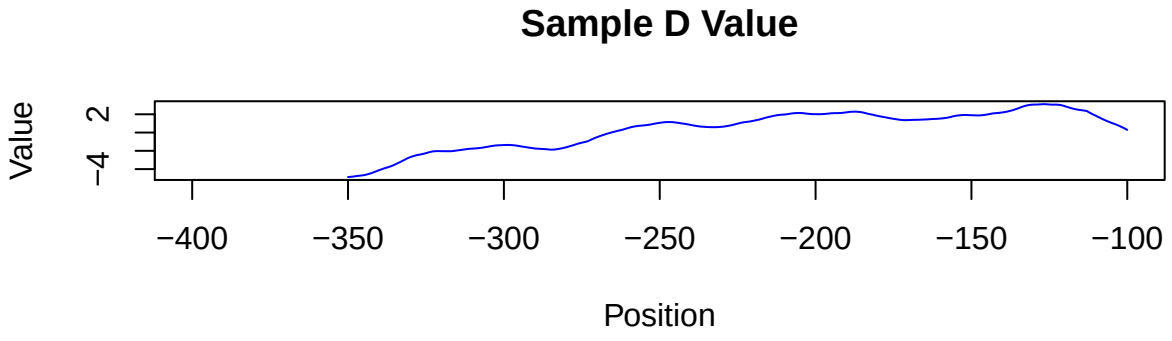
```
D <- read.table("Data/D.txt", header=T)
E1 <- read.table("Data/E1.txt", header=T)
E2 <- read.table("Data/E2.txt", header=T)
promoters <- read.table("Data/predicted_promoters.txt", header=T)
```

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

Muestrear una secuencia.

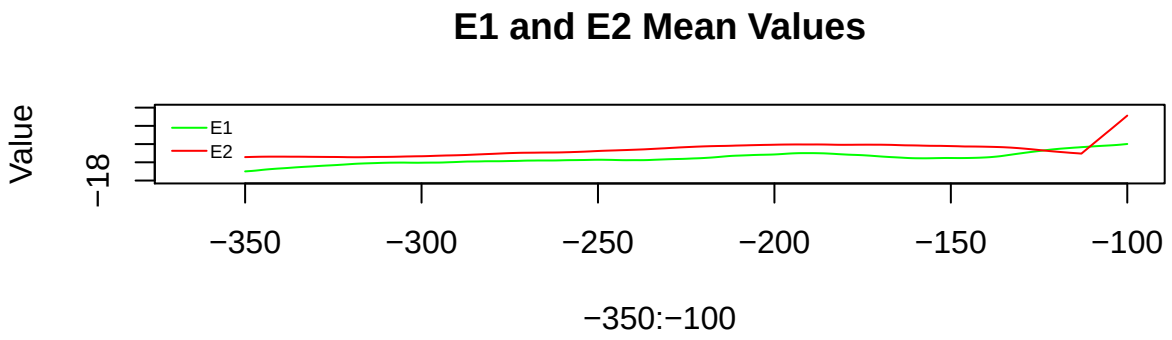
```
par(mfrow=(c(2,1)))
x <- sample(1:84,1)
plot(x=-350:-100,E1[x,], col="green", type="l",xlab = "Position", ylab="Value",main = "Sample E1 and E2
lines(x = -350:-100,E2[x,],col="red")
legend("topleft", c("E1","E2"),lty=1,col=c("green","red"),bty="n",cex=0.5, y.intersp = 1,x.intersp=0.2)
plot(x=-350:-100,D[x,], col="blue", main = "Sample D Value", type="l",xlab = "Position",ylab="Value",xl
```

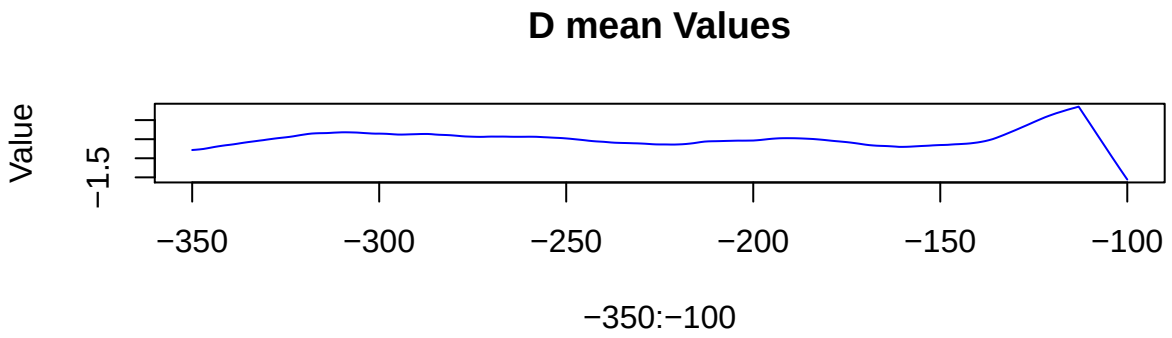




Promedio de todas las secuencias.

```
par(mfrow=(c(2,1)))
plot(x=-350:-100,apply(E1,2,mean), col="green", type="l",xlim=c(-365,-100),ylim=c(-18,-14), ylab="Value
legend("topleft", c("E1","E2"),lty=1,col=c("green","red"),bty="n",cex=0.6, y.intersp = 1,x.intersp=0.2)
lines(x = -350:-100,apply(E2,2,mean),col="red")
plot(x=-350:-100,apply(D,2,mean), col="blue", main = "D mean Values", type="l",ylab="Value")
```



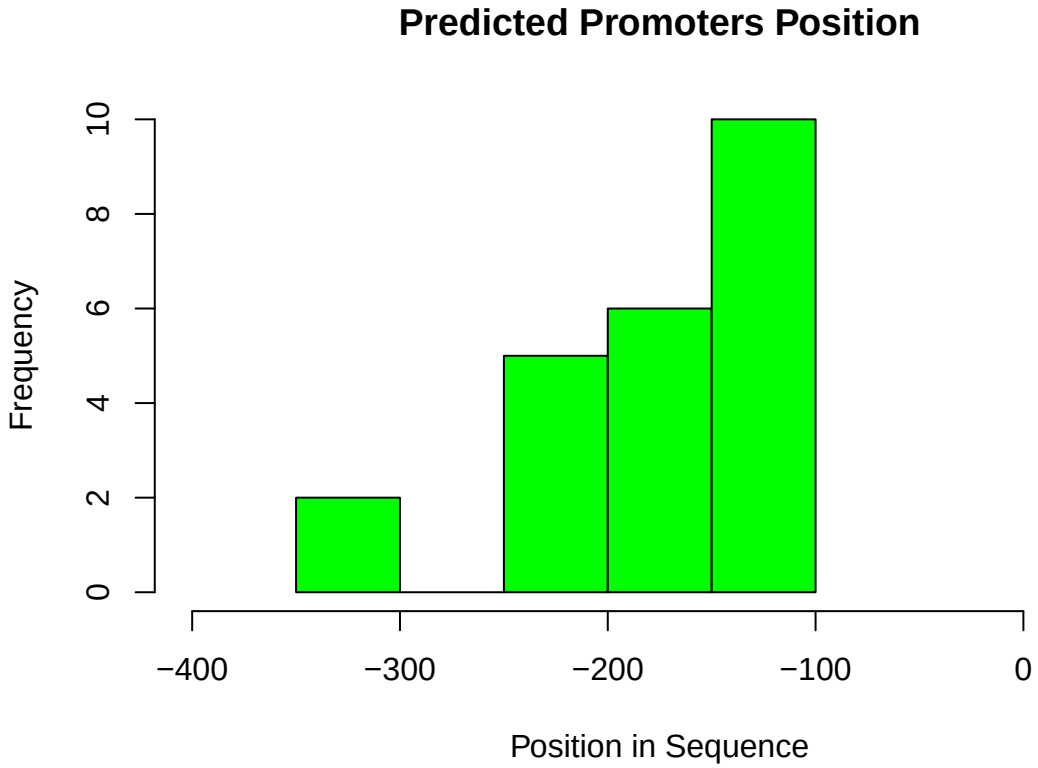


3) Predecir promotores en todas las secuencias del fichero K12_400_50_sites. *Correr Perl*

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?

```
par(mfrow=c(1,1))

hist(promoters$Pos-400,xlim = c(-400,50),ylab="Frequency",xlab="Position in Sequence", main="Predicted
```



Además del método para obtener falsos positivos descrito en el artículo basado en buscar overlaps de 200 nucleótidos en la región -150 a 50 de la secuencia analizada, con RSAT, podrían realizarse diversas pruebas mejorando únicamente los valores de sensibilidad (que reflejará la capacidad del estimador para detectar verdaderos positivos, indicando de la misma forma su comportamiento con los falsos positivos) y con la herramienta matrix-scan podemos buscar los sitios de pegado de los factores de transcripción, y buscar si estos corresponden a las regiones promotoras encontradas por el algoritmo.

En la figura 7 del artículo, puede observarse que los valores de los resultados mejoran al ampliar el tamaño de la ventana de selección.