Practica2

Carmina Barberena Jonas

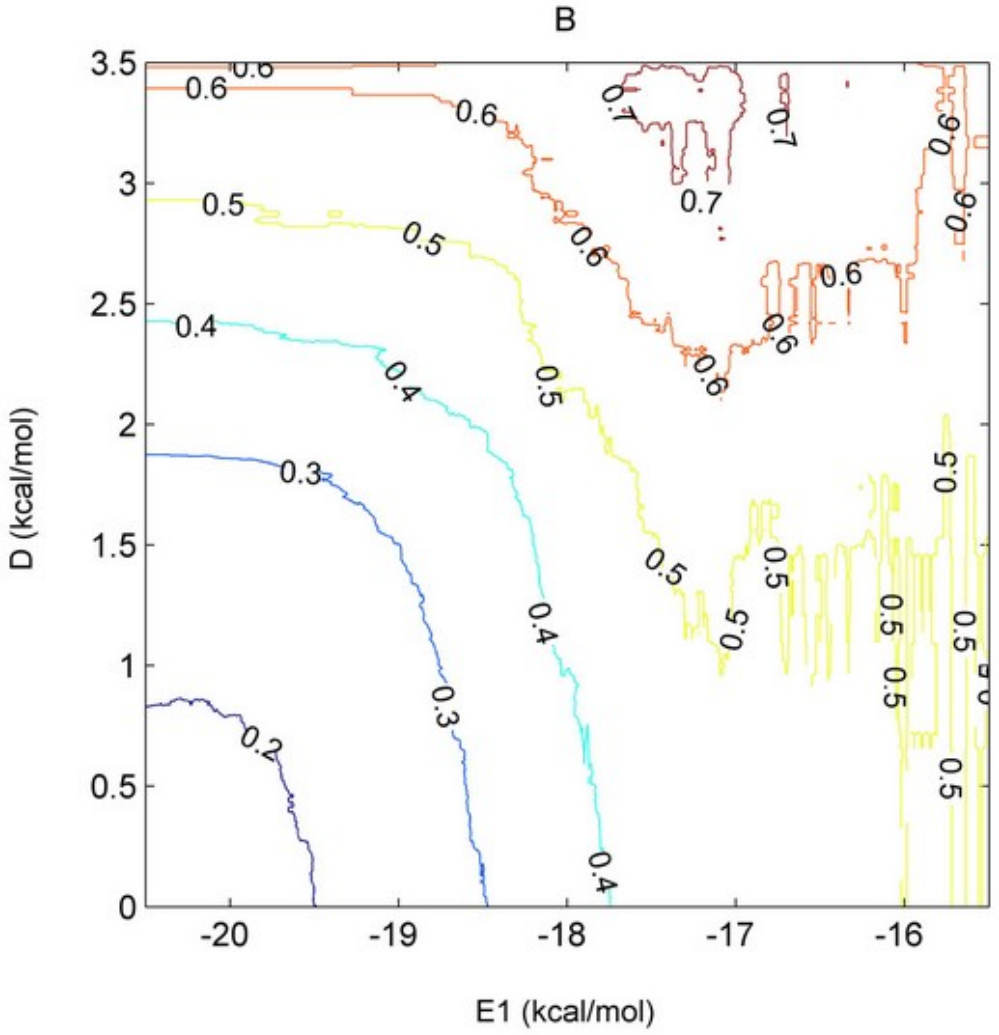
6 de marzo de 2016

Practica 2

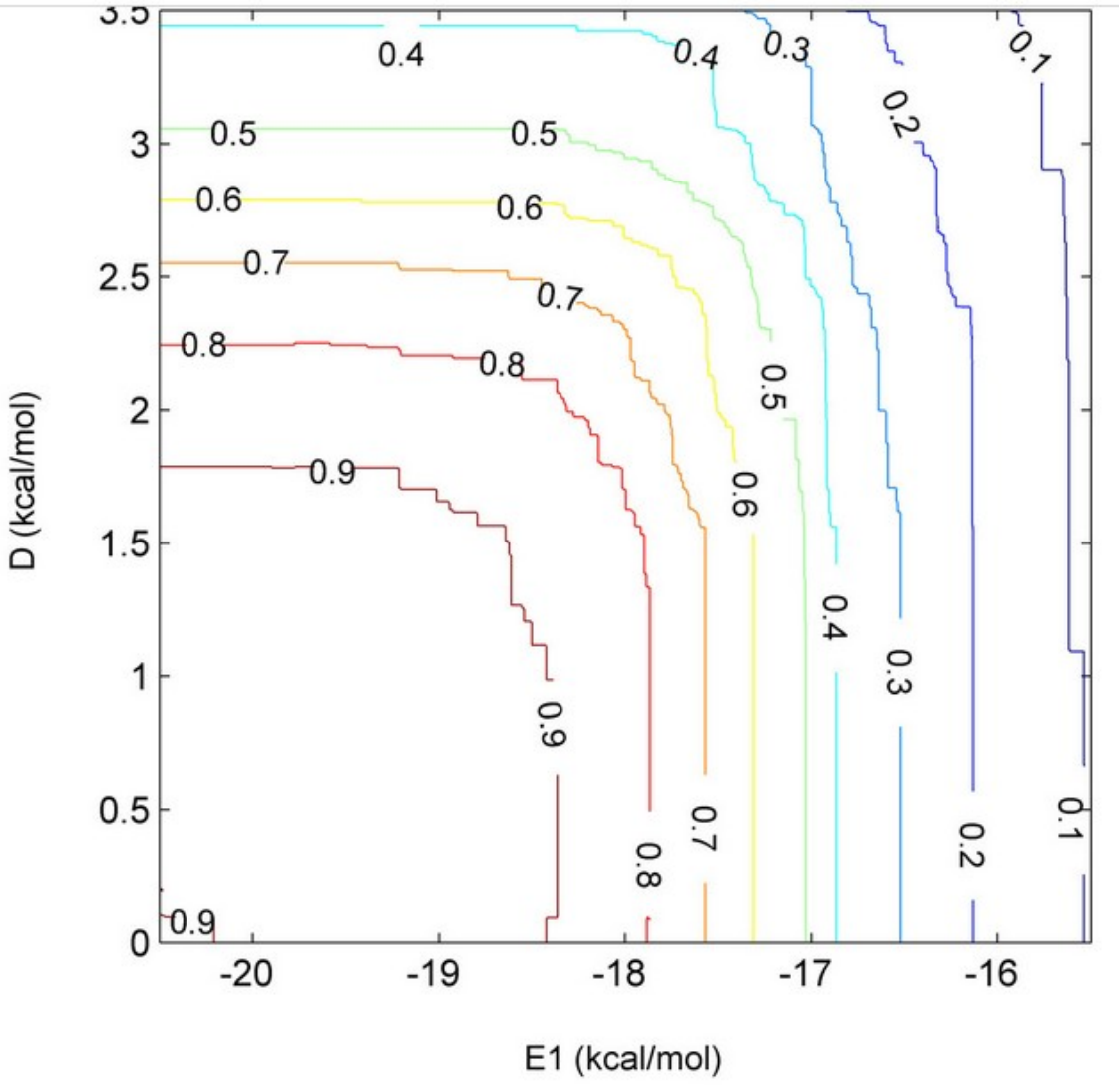
Carmina Barberena Jonas

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.

Valores de Corte 1 y 2 Sensitividad



Precisión



Tabla

Table 1

The number of false positives obtained for different levels of sensitivity.

Sensitivity	Cut-off for D	Cut-off for E1 (kcal/mole)	Frequency of false positives	
			FP (1/nt)[a]	FP (1/nt)[b]
0.13	3.4	-15.99	1/16214	1/261000
0.22	3.4	-16.7	1/11350	1/130500
0.32	3.3	-17.1	1/8407	1/65250
0.40	3.3	-17.55	1/6486	1/29000
0.50	2.76	-17.53	1/3914	1/13737
0.60	2.45	-17.64	1/2467	1/7250
0.70	2.35	-18.07	1/1621	1/2747
0.81	1.9	-18.15	1/1086	1/1878
0.90	0.97	-18.37	1/572	1/967

Se escojieron los valores de cortes mas estrictos para poder tener mas confianza en las predicciónes que se obtengan. Con estos valores esperamos que aunque no tengamos una gran cantidad de promotores sean muy posiblemente sitios reales.
my $cuttoffE1=-15.99; my $cuttoffD=3.4;

Codigo en Perl

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

# prog1.1
# Bruno Contreras-Moreira /Carmina Barberena Jonas
# 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 %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 #Correción para secuencias palindromicas
   '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";
my %SEQUEN ; #Declarar un hash vacio para almacenar secuencias
my %DeltaG ; #Variable para guardar el DeltaG
my %E1; #Guardar el E1
my %E2; #Guardar el E2
my %D; #Guardar el Distance
my %E1n;
my %E2n;
```

```
my %Dn;
open(SEQ, $infile) || die "# cannot open input $infile : $!\n";
while(<SEQ>)
{
   if(/^(b\d{4}) \\ ([ATGC]+)/)
   {
      my ($name,$seq) = ($1,$2);
      #printf("sequence %s (%d nts)\n",$name,length($seq)); #Prueba
      $SEQUEN{$name} = $seq; # LLegar hash usando como llave el nombre

   }
}

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) = @_;
   my ($DNAstep,$nt,$dG,$total_dG) = ('','',0,0);
   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 $n=0;$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;
   }
   #print "el total deltaG es $total_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 (substr($seq,0,7) eq substr(reverse (complement($seq)),0,7)){ #Checar
la simetria de la secuencia
      $total_dG += ((1000*$NNparams{'sym'}{'H'})- #Modificar el valor
               ($tK*$NNparams{'sym'}{'S'}))
               / 1000;
      }


   return $total_dG;
}
my $con;
my $cuttoffE1=-15.99;
my $cuttoffD=3.4;
foreach my $llave (keys %SEQUEN){
   $con=1;
   # Obtener los deltas correspondientes a cada ventana.
   for (my $n=0; $n<=((length($SEQUEN{$llave}))-15);$n++){
      #print $n;
      my $window = substr($SEQUEN{$llave},$n,15);
      #print "$llave = $SEQUEN{$llave}\n";
      $DeltaG{$llave."-".$n}=duplex_deltaG($window,$T); #LLamar a la
subrutina ṕor cada ventada de cada secuencia
      }

   for (my $j=0; $j<=((length($SEQUEN{$llave}))-215);$j++){ # Se resta
200 por secuencia y 15 por ventandas

      #####Optener el valor de E1####
      for (my $m=$j; $m<=$j+49; $m++) {
         $E1{$llave."-".$j}+= $DeltaG{$llave."-".$m};
         #$con ++;
      }
      $E1{$llave."-".$j}=$E1{$llave."-".$j}/50 ;
      ####Optener el valor de E2 ####
      for (my $h=$j+99; $h<=$j+199; $h++) {
         $E2{$llave."-".$j}+= $DeltaG{$llave."-".$h} ;
      }
      $E2{$llave."-".$j}=$E2{$llave."-".$j}/100;
      $D{$llave."-".$j}=$E1{$llave."-".$j}-$E2{$llave."-".$j} ;
   }

}
#########Promedios de E1 E2 y D por N ##############
for (my $n=0; $n<=(235);$n++){
```

```
   foreach my $llave (keys %SEQUEN){
      $E1n{$n}+=$E1{$llave."-".$n};
      $E2n{$n}+= $E2{$llave."-".$n};
      $Dn{$n}+= $D{$llave."-".$n};
   }
}
###############Imprimir las variables""""""""""
# foreach my $llave (keys %SEQUEN){
# $con=1;
# for (my $j=0; $j<=((length($SEQUEN{$llave}))-215);$j+=24){ # Se resta
200 por secuencia y 15 por ventandas (Se suma 24 por que se toma como 1 )
   # if ($D{$llave."-".$j}>$cuttoffD && $E1{$llave."-".$j}>$cuttoffE1){
         # print $llave."-".$con."\t Pos".($j)."\t Inicio".($j-350)."\t Fin".($j-
300) ."\n";
         # #$SeqProm{mash}=($llave."-".$con, {'Pos',$j,'Inicio',$j+49,'Fin',
$j+99});
         # $con ++;
      # }
   # }
# }
# for (my $n=0; $n<=(235);$n++){
   # print $E1{"b0972-".$n}."\t".$E2{"b0972-".$n}."\t".$D{"b0972-".
$n}."\n";
# }



# foreach my $llave (keys %E1n){
   # print $llave."\t".$E1n{$llave}."\n";
   # }
# foreach my $llave (keys %E2n){
   # print $llave."\t".$E2n{$llave}."\n";
   # }
# foreach my $llave (keys %Dn){
   # print $llave."\t".$Dn{$llave}."\n";
   # }

# foreach my $llave (keys %E1){
   # print $llave=$E1{$llave}."\n";
   # }
# foreach my $llave (keys %E2){
   # print $llave =$E2{$llave}."\n";
   # }
# foreach my $llave (keys %D){
   # print $llave =$D{$llave}."\n";
   # }
```

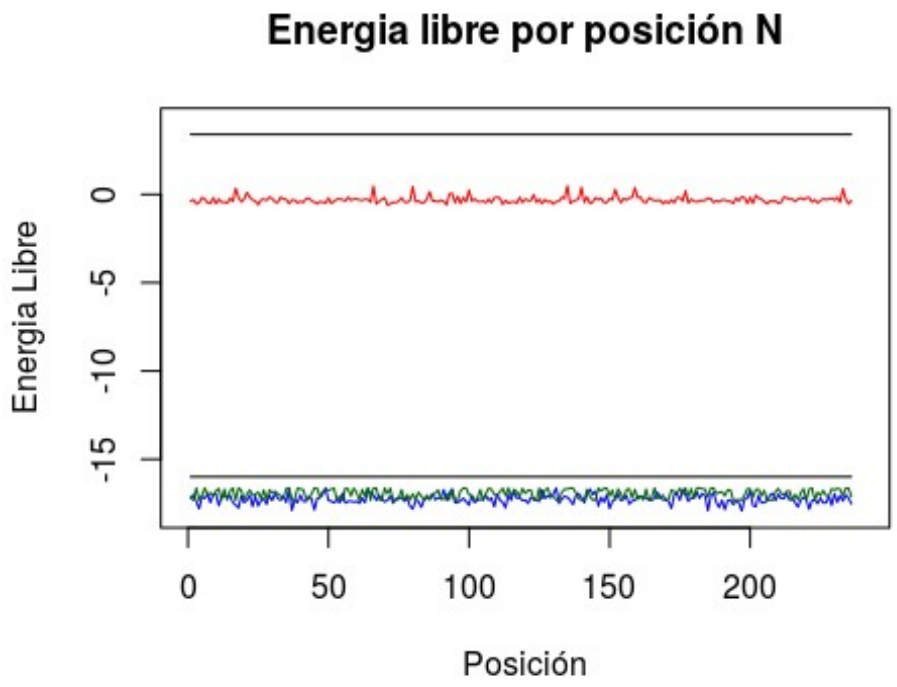
2 Diseñar una figura donde se muestra gráficamente D, E1 y E2 para una posición n. #Graficas de E1, E2 y D Primeramente se obtienen

los valores obtenidos en el script de perl y se almacenan en variables. Ademas de agregar los cortes.

```
E1n<-read.table(file="/home/cbarbere/Bioinfo/Bruno/E1n")
E2n<-read.table(file="/home/cbarbere/Bioinfo/Bruno/E2n")
Dn<-read.table(file="/home/cbarbere/Bioinfo/Bruno/Dn")
E1np<-E1n$V2/84
E2np<-E2n$V2/84
Dnp<-Dn$V2/84
b0972<-read.table(file="/home/cbarbere/Bioinfo/Bruno/b0972")
cutoffD<-rep(3.4,236)
cutoffE<-rep(-15.99,236)
```

A contuación graficaremos el promedio de E1, E2 y D por posición. Marcando los puntos de corte

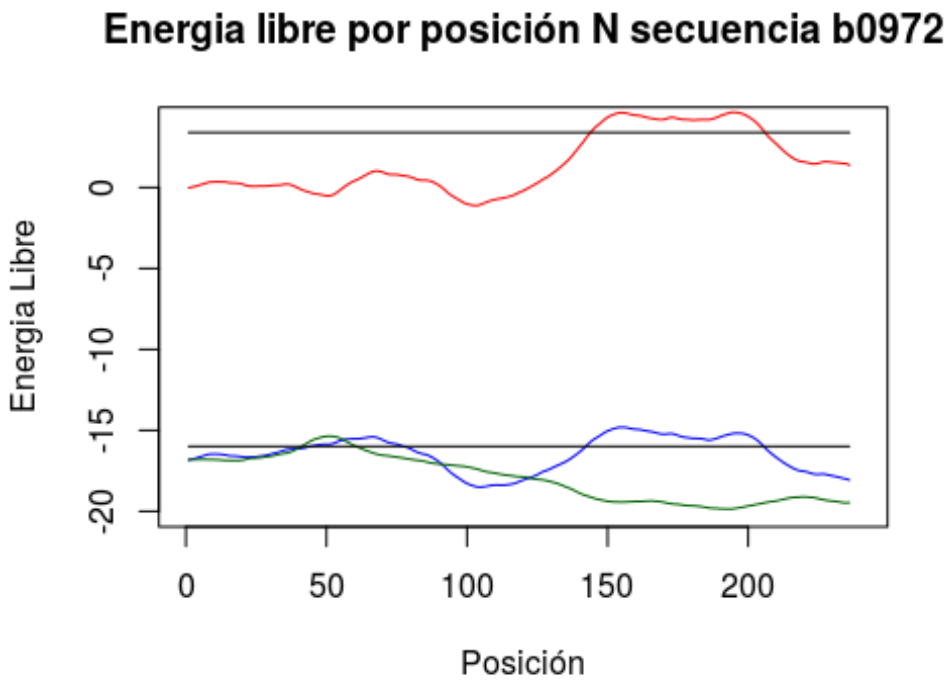
```
plot(Dnp, xlim=c(0,240), ylim=c(-18,4), type="l",
    xlab="Posición", ylab="Energia Libre",col="red",
    main=("Energia libre por posición N"))
lines(E1np, col="blue")
lines(E2np, col="darkgreen")
lines(cutoffD)
lines(cutoffE)
```



Como podemos ver en el promedio ninguna posición supero los umbrales marcados,lo que nos indica que las regiones promotoras se encuentran distribuidas de manera difernte a lo largo de las secuencias. Por esto

usaremos una secuencia unicamente para poder ver el comportamineto individual.

```
b0972<-read.table(file="/home/cbarbere/Bioinfo/Bruno/b0972")
plot(b0972$V3, xlim=c(0,240), ylim=c(-20,4), type="l",
    xlab="Posición", ylab="Energia Libre",col="red",
    main=("Energia libre por posición N secuencia b0972"))
lines(b0972$V1, col="blue")
lines(b0972$V2, col="darkgreen")
lines(cutoffD)
lines(cutoffE)
```

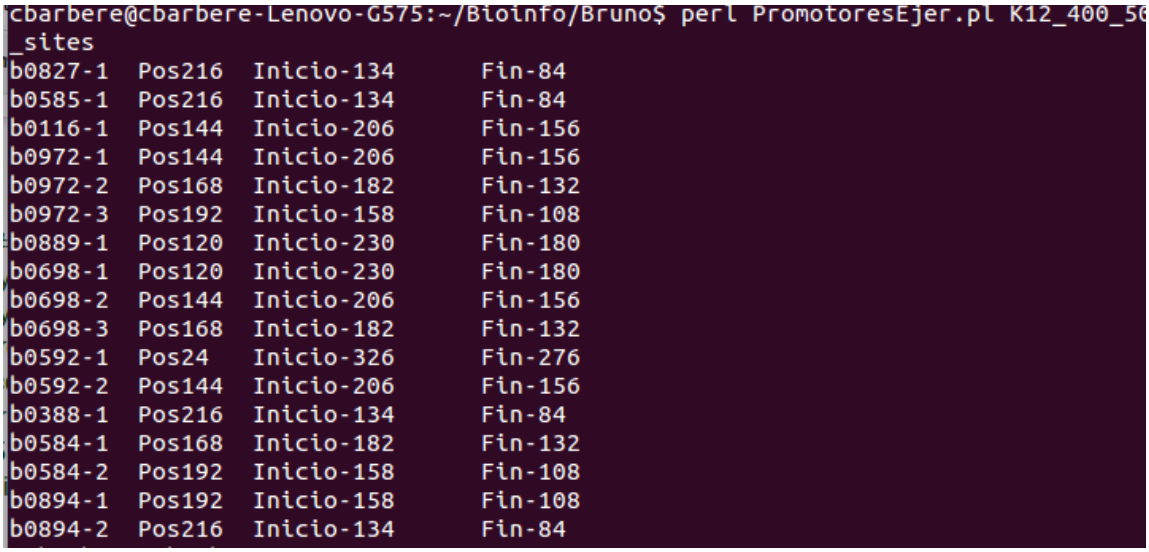


En esta grafica podemos observar como los valores sobrepasan el corte seleccionado tanto para D como para E1. Podemos identificar una región grande donde se cumplen ambas condiciones, esto cerca de las posiciones -150 a -200.
##Predicción de Promotores

```
######Predicción de Promotores######
my $cuttoffE1=-15.99;
my $cuttoffD=3.4;
foreach my $llave (keys %SEQUEN){
$con=1;
for (my $j=0; $j<=((length($SEQUEN{$llave}))-215);$j+=24){ # Se resta 200 por secuencia y 15 por ventandas (Se suma 24 por que se toma como 1 )
   if ($D{$llave."-".$j}>$cuttoffD && $E1{$llave."-".$j}>$cuttoffE1)
```

```
{ #Contenga las 2 condiciones
        print $llave."-".$con."\t Pos".($j)."\t Inicio".($j-350)."\t Fin".($j-
300) ."\n";
        $con ++;
      }
   }
}
```

Resultado

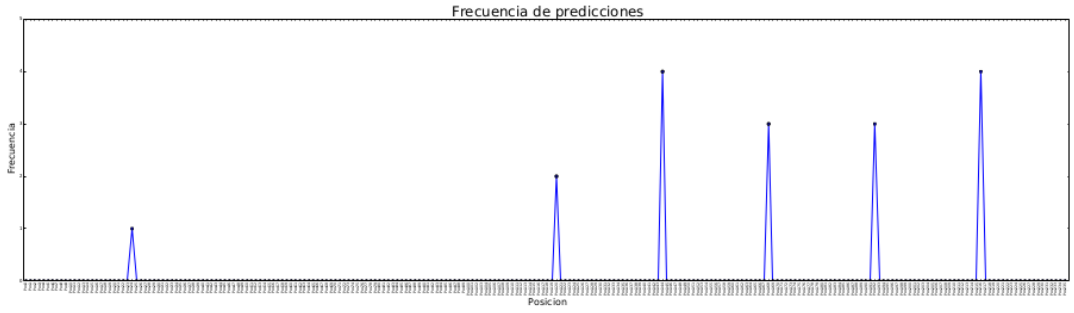


4 Graficar con qué frecuencia se predicen promotores en el intervalo -400,50. Con un breve comentario de los resultados es suficiente. Se les ocurre una manera de validar sus resultados, y calcular la tasa de FPs, usando RSAT::matrix-scan? Procegiremos a comprobar que las regiones promotoras se encuentran distribuidas a lo largo de la secuencia con un grafico de frecuencias.

Codigo en Phyton

```
get_ipython().magic(u'matplotlib inline')

import numpy as np
import matplotlib.pyplot as plt
import natsort

dic={}
with open('/Users/MainU/Desktop/Promotores.jpg','r') as f:
    for line in f:
        key=line.split('\t')[1].replace(' ','')
        if key in dic:
            dic[key]=dic[key]+1
        else:
            dic[key]=1
```

```
for i in range(236):
    newKey='Pos'+str(i)
    if newKey not in dic:
        dic[newKey]=0

sortkeys=natsort.natsorted(dic.keys(), key=lambda y: y.lower())

fig=plt.figure()
fig.set_size_inches(40, 10)
ax = plt.axes()
ax.set_xlim(-0.5,len(sortkeys))
ax.set_ylim(0,5)
ax.set_xticks([x+0.01 for x in range(len(sortkeys))])
#ax.set_xticks([0.01,1.01,2.01,3.01,4.01,5.01,6.01,7.01,8.01,9.01, 10.01,
11.01])
ax.set_xticklabels(sortkeys,rotation=90,fontsize=8)
values=[dic[x] for x in sortkeys]
plt.plot(values,color="blue", marker='o')
ax.set_ylabel('Frecuencia', fontsize=20)
ax.set_xlabel('Posicion', fontsize=20)
plt.title('Frecuencia de predicciones', fontsize=30)
plt.show()
fig.savefig('/Users/MainU/Desktop/frecuencias.pdf')
```



Por usar valores de corte tan estrictos no pudimos ver con claridad la frecuencia ya que se tienen muy pocas predicciónes. Matrix-scan nos permite buscar en una secuencia los lugares donde se uniria matrix, por lo que podriamos tomar las regiones que nos predice nuestro programa y hacer una busqueda en las regiones -400 50 para ver a cuentos se esta uniendo y compararlo con una secuencia al azar. Así podriamos ver los genes que podria estar regulando.