```
#!/bin/perl

use List::Util qw[min max sum];


sub TDD
{
# hash of arrays with thermodynamic parameters for DNA/DNA duplex
# hash keys are respective pairs
# first array element is enthalpy (dH)
# second array element is enthropy (dS)
my %data_hash = (
'AA' => [7.9, 0.0222],
'TT' => [7.9, 0.0222],
'AT' => [7.2, 0.0204],
'TA' => [7.2, 0.0213],
'CA' => [8.5, 0.0227],
'TG' => [8.5, 0.0227],
'GT' => [8.4, 0.0224],
'AC' => [8.4, 0.0224],
'CT' => [7.8, 0.021],
'AG' => [7.8, 0.021],
'GA' => [8.2, 0.0222],
'TC' => [8.2, 0.0222],
'CG' => [10.6, 0.0272],
'GC' => [9.8, 0.0244],
'GG' => [8.0, 0.0199],
'CC' => [8.0, 0.0199]
);

# input parameters for the script
my ($seq, $start, $window, $max_length) = @_; #sequence file, calculation
start position, window size, max length
my $stop = 100000000000;         # calculation end position
my $step = 1;                    # step

#cutting the seq as much as we need
my $Cut = $start - 0;
$seq =~ s/^.{$Cut}//;

# some variables we are gonna use later
# define them here as globals to avoid scope issues
my $S;                       # entropy
my $H;                       # entalpy
my $G;                       # free energy
my $roundG;                  # rounded free energy, used for the output
my $input_seq;        # input sequence after stiching the lines
my $current_pos; # current position within the sequence
my $N_character = 'N';
my $x_character = 'x';

my @value_array;
```

```
for (my $i = 0; $i < $MaxLength; ++$i )
{
  push(@value_array, "0");
}

#print $seq."\n";
my $i;
$counter = 0;
for ($i = 0; $i <= length($seq) - $window; $i+= $step) # incerement by
the given step
{
      # zero the variables
      $S = 0;
      $H = 0;
      $G = 0;

      $counter += $step;
      # define a subsequnce based on the window size
      my $sub_seq = substr($seq,$i,$window);

      # removes sequences with Ns in them
      if ($sub_seq =~ /$N_character/) { next; }

      # removes sequences with xs in them
      if ($sub_seq =~ /$x_character/) { next; }

      # extract all the nearest neighbour pairs
      for (my $j = 0; $j < length($sub_seq); $j++)
            {
                  my $pair = substr($sub_seq,$j,2);
                  $pair = uc($pair);          # convert to upper case

                  # get respective values from the hash
                  $H += $data_hash{$pair}->[0];
                  $S += $data_hash{$pair}->[1];

                        # calculate free energy
                        if ( $S != 0 )
                        {
                              $G =$H*(1-310/($H/$S));
                        }
                        else
                        {
                              #in case we somehow didn't manage to avoid
N characters
                              $G = 0;
                        }
            }

      $roundG = sprintf "%.2f",$G;     # round the value
      $current_pos = $start  + $counter + $window/2 -1;      # calculate
where we are now. If window size is an odd number, add 0.5
```

```
      if ( $window % 2 != 0 )
      { $current_pos += 0.5; }
      #print "$current_pos\t$roundG\n";                # print output on
screen
      $value_array[$current_pos] = $roundG;

      # $counter += $step;
}
# calculate not parsed string at the end of $seq to concat with the next
iteration
#$seq = substr($seq, $i, length($seq) - $i);
#print "@value_array";
return @value_array;
}

sub TRD
{
# hash of arrays with thermodynamic parameters for RNA/DNA duplex
# hash keys are respective pairs
# first array element is enthalpy (dH)
# second array element is enthropy (dS)
my %data_hash = (
'AA' => [7.8, 0.0219],
'TT' => [11.5, 0.0364],
'AT' => [8.3, 0.0239],
'TA' => [7.8, 0.0232],
'CA' => [9.0, 0.0261],
'TG' => [10.4, 0.0284],
'GT' => [7.8, 0.0216],
'AC' => [5.9, 0.0123],
'CT' => [7.0, 0.0197],
'AG' => [9.1, 0.0235],
'GA' => [5.5, 0.0135],
'TC' => [8.6, 0.0229],
'CG' => [16.3, 0.0471],
'GC' => [8.0, 0.0171],
'GG' => [12.8, 0.0319],
'CC' => [9.3, 0.0232]
);

# input parameters for the script
my ($seq, $start, $window, $max_length) = @_; #sequence file, calculation
start position, window size, max length
my $stop = 100000000000;         # calculation end position
my $step = 1;                    # step

#cutting the seq as much as we need
my $Cut = $start - 0;
$seq =~ s/^.{$Cut}//;

# some variables we are gonna use later
# define them here as globals to avoid scope issues
```

```
my $S;                      # entropy
my $H;                      # entalpy
my $G;                      # free energy
my $roundG;                 # rounded free energy, used for the output
my $input_seq;        # input sequence after stiching the lines
my $current_pos; # current position within the sequence
my $N_character = 'N';
my $x_character = 'x';

my @value_array;
for (my $i = 0; $i < $MaxLength; ++$i )
{
  push(@value_array, "0");
}

#print $seq."\n";
my $i;
$counter = 0;
for ($i = 0; $i <= length($seq) - $window; $i+= $step) # incerement by
the given step
{
      # zero the variables
      $S = 0;
      $H = 0;
      $G = 0;

      $counter += $step;
      # define a subsequnce based on the window size
      my $sub_seq = substr($seq,$i,$window);

      # removes sequences with Ns in them
      if ($sub_seq =~ /$N_character/) { next; }

      # removes sequences with xs in them
      if ($sub_seq =~ /$x_character/) { next; }

      # extract all the nearest neighbour pairs
      for (my $j = 0; $j < length($sub_seq); $j++)
            {
                  my $pair = substr($sub_seq,$j,2);
                  $pair = uc($pair);          # convert to upper case

                  # get respective values from the hash
                  $H += $data_hash{$pair}->[0];
                  $S += $data_hash{$pair}->[1];

                  # calculate free energy
                  if ( $S != 0 )
                  {
                        $G =$H*(1-310/($H/$S));
                  }
                  else
```

```
                {
                        #in case we somehow didn't manage to avoid N
characters
                        $G = 0;
                }
           }

      $roundG = sprintf "%.2f",$G;     # round the value
      $current_pos = $start  + $counter + $window/2 -1;      # calculate
where we are now. If window size is an odd number, add 0.5
      if ( $window % 2 != 0 )
      { $current_pos += 0.5; }
      #print "$current_pos\t$roundG\n";                # print output on
screen
      $value_array[$current_pos] = $roundG;

      # $counter += $step;
}
# calculate not parsed string at the end of $seq to concat with the next
iteration
#$seq = substr($seq, $i, length($seq) - $i);
#print "@value_array";
return @value_array;
}


sub FileParser
{

#files names
 my ($infile, $outfile, $start, $window, $MaxLength, $bool, $avg_file,
$empty) = @_;

 #lines for working
 my $line;
 my $line1 = '';
 my $avoider = 0;
 my $Delimiter = "\t";
 my $Filler = " ";

 #opening files
 open( IN, $infile ) or die "Can't open input file.\n";
 open( OUT, ">$outfile" ) or die "Can't open output file.\n";
 open( AVG, ">$avg_file" ) or print "Couldn't open avg file.\n";
 open( EMPTY, $empty ) or print "Couldn't open file with empty values.\n";


 my @name_array;

 my @seq_array;

 while ( $line = <IN> )
```

```
 {
   if ( $line =~ /\>/ )
   {
     chomp($line);
     push( @name_array, $line);
     if ( $avoider == 0 ) { $avoider = 1; next; }
     push( @seq_array, $line1);
     $line1 = '';
     next;

   }
   chomp ($line);
   $line1 = $line1.$line;
 }
 push ( @seq_array, $line1 );#last sequence


if ( $MaxLength <= 0 )
{
 print "Maximum lenght can't be less than 0 or empty";
}

my %Empty;

while ( <EMPTY> )
{
      my $EmptyName;
      my $EmptyValue;

      ( $EmptyName, $EmptyValue ) = split( /\s/, $_);

      $Empty{ $EmptyName } = $EmptyValue;
}

for ( my $index = 0; $index <= $#name_array; ++$index )
{
      if ( $seq_array[$index] !~/ / and $seq_array[$index] !~/N/ ) #if there
is " " in sequence we assume there is no sequence at all
      {
            my @value_array;
            #we will cut "length of sequence - MaxLength" symbols at the
end of sequence,
            #so in the end we will have at most

            my $CutLength = length($seq_array[$index]) - $MaxLength;
            #print $CutLength;

            #actual cutting of the string
            $seq_array[$index] =~ s/.{$CutLength}$//;

            if ( $bool =~ /RD/ )
            {
```

```
                @value_array = TRD( $seq_array[$index], $start, $window,
$Max_Length );
            }
            elsif ( $bool =~ /DD/ )
            {
                @value_array = TDD( $seq_array[$index], $start, $window,
$Max_Length );

            }
            elsif ( $bool =~ /T4/ )
            {
                my @DD2 = { 0, };
                my @RD1 = TRD( $seq_array[$index], 2, 9, $Max_Length );
                my @RD2 = TRD( $seq_array[$index], 3, 8, $Max_Length );
                my @DD1 = TDD( $seq_array[$index], 1, 12, $Max_Length );
                push ( @DD2, TDD( $seq_array[$index], 0, 12, $Max_Length
) );
                for ( my $i = 0; $i < $MaxLength; ++$i )
                {
                    $value_array[$i] = - $RD1[$i] - $DD1[$i] + $RD2[$i]
+ $DD2[$i];
                    $value_array[$i]= sprintf "%.2f",$value_array[$i];

                }

                my $counter = 0;
                for ( my $i = $#value_array; $i > 0 ; --$i )
                {
                    if ( $counter >= 2 )
                    {
                        last;
                    }
                    if ( $value_array[$i] != 0 )
                    {
                        $value_array[$i] = 0;
                        ++$counter;
                    }
                }

            }
            elsif ( $bool =~ /T2/ )
            {
                my @RD = TRD( $seq_array[$index], 0, 9, $Max_Length );
                my @DD = TDD( $seq_array[$index], 0, 9, $Max_Length );
                for ( my $i = 0; $i < $MaxLength; ++$i )
                {
                    $value_array[$i] = $DD[$i] - $RD[$i];
                    $value_array[$i]= sprintf "%.2f",$value_array[$i];
                }

            }
```

```
            while ( $#value_array < ($MaxLength - 1) )
            {
                  push ( @value_array, 0 );
            }

            #here we do the empting if there is need of one
            if ( $Empty{ $name_array[$index] } )
            {
                  my $TmpEmpty = 150 - $Empty{ $name_array[$index] };
                  $seq_array[$index] =~ s/^.{$TmpEmpty}//;
                  #print "Seq: $seq_array[$index]\n";

                  my $TmpStart = $start + $window/2;
                  if ( $window % 2 != 0 )
                  {
                        $TmpStart += 0.5;
                  }

                  for ( my $i = 0; $i < ($TmpEmpty + $TmpStart) ; ++$i )
                  {
                        $value_array[$i] = 0;
                  }
            }

            my $counter = 0;
            my $Avg = 0;
            my $Min = 100;
            my $Max = 0;
            print OUT "$name_array[$index]$Delimiter";
      my $seqLength = length($seq_array[$index]);
            for (my $i = 1; $i <= $#value_array ; ++$i )
            {
        if ( $seqLength > $MaxLength ) {
          $seqLength = $MaxLength
        }
                  if ( ($value_array[$i]!= 0)  or (($i > ($window / 2)) and
($i < $seqLength - ($window / 2))))
                  {
                        #we are calculating min, max and avg
                        ++$counter;

                        $Avg += $value_array[$i];

                        if ( $value_array[$i] < $Min )
                        {
                              $Min = $value_array[$i];
                        }

                        if ( $value_array[$i] > $Max )
                        {
                              $Max = $value_array[$i];
                        }
```

```
                        $value_array[$i] =~ s/\./,/;
                        print OUT "$value_array[$i]$Delimiter";

                }
                else
                {
                        print OUT "$Filler$Delimiter";
                }
        }
        print OUT "$seq_array[$index]\n";


        if ( $counter != 0 )
        {
                $Avg = $Avg / $counter;
                $Avg = sprintf "%.2f",$Avg;
        }
        else
        {
                $Avg = 0;
        }

        $Min =~ s/\./,/;
        $Max =~ s/\./,/;
        $Avg =~ s/\./,/;
        #we print Name, Min, Max, Avg and Sequence in the file
        print AVG
"$name_array[$index]$Delimiter$Min$Delimiter$Max$Delimiter$Avg$Delimit
er$seq_array[$index]\n";

    }
}

}


my $ConfigFile = shift;

open ( CONFIG, $ConfigFile) or die "Can't open file with configuration.\n";

#work variables
my $ConfigName;
my $ConfigValue;

#map we use for storing configurations
my %Configuration;


while ( <CONFIG> )
{
```

```
 ($ConfigName, $ConfigValue) = split (/\=/, $_);
 $Configuration{ $ConfigName } = $ConfigValue;
}
close( CONFIG );

print "We are going to do: $Configuration{\"Use\"}";

FileParser( $Configuration{"InputFile"},
            $Configuration{"OutputFile"},
            $Configuration{"Start"},
            $Configuration{"Window"},
            $Configuration{"MaxLength"},
            $Configuration{"Use"},
            $Configuration{"Average"},
            $Configuration{"EmptyFile"}
          );
```