University of Richmond
UR Scholarship Repository

Honors Theses | Student Research

1999

Method for identification of origins of replication and genes regulated by DnaA in bacteria

Olga G. Troyanskaya
University of Richmond

Follow this and additional works at: https://scholarship.richmond.edu/honors-theses

 Part of the Biology Commons, Computer Sciences Commons, and the Mathematics Commons

Recommended Citation

Troyanskaya, Olga G., "Method for identification of origins of replication and genes regulated by DnaA in bacteria" (1999). *Honors Theses*. 966.
https://scholarship.richmond.edu/honors-theses/966

This Thesis is brought to you for free and open access by the Student Research at UR Scholarship Repository. It has been accepted for inclusion in Honors Theses by an authorized administrator of UR Scholarship Repository. For more information, please contact scholarshiprepository@richmond.edu.

UNIVERSITY OF RICHMOND LIBRARIES
3 3082 00688 8530

Biology
Tro

Method for identification of origins of replication and genes regulated by DnaA in bacteria

Olga G. Troyanskaya

Honors Thesis

In

Departments of Biology and Mathematics and Computer Science

University of Richmond

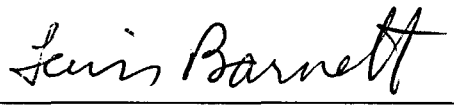
Richmond, VA

April 23, 1999

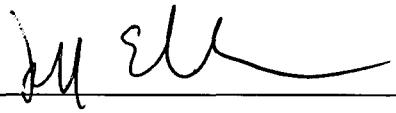
Advisors: Jeff Elhai*, Lewis Barnett+, Steven Salzberg^.

*Department of Biology, University of Richmond
+Department of Mathematics and Computer Science, University of Richmond
^Department of Bioinformatics, The Institute for Genomic Research, MD

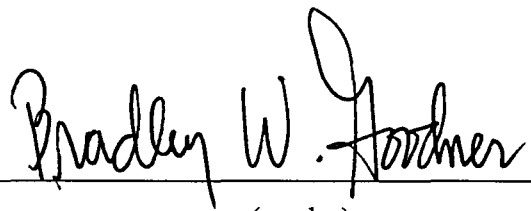
This paper is part of the requirements for an interdisciplinary honors program in biology and computer science. The signatures below, by the advisors, a departmental reader, and a representative of the departmental honors committee, demonstrate that Olga Troyanskaya has met all the requirements needed to receive honors in biology and computer science.



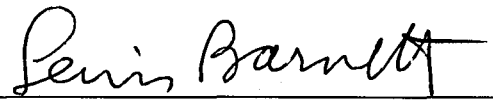
(advisor)



(advisor)



(reader)



(honors committee representative)

Abstract

The study is focused on developing computer programs to identify origin of DNA replication based on analysis of total bacterial genomes, scoring regions for number of DnaA binding sites, AT content, DNA adenine methylase boxes, and integration host factors binding sites. The programs were tested on cyanobacterium *Synechocystis*, and several potential origins were identified. However, no one definite region could be located. Currently, software is being developed to analyze common motifs around the origins of all bacteria with known origins. Genes whose transcription could be regulated by DnaA were identified by searching for DnaA boxes preceding promoter regions.

Introduction

All organisms face the problem of DNA replication, which happens once every cell cycle before cell division. DNA replication must be under strict control with respect to cell division or else daughter cells will either lack DNA or have too much DNA. In mammals, loss of control over the cell cycle DNA replication is a major event leading to cancer. In many bacteria, DNA replication is initiated at a unique site on the chromosome, the origin of replication (Skarstad and Boye, 1994).

DnaA is a central protein in DNA replication initiation in bacteria. *E. coli* cells that lack functional DnaA protein and don't have secondary compensatory mutations are not viable (Skarstad and Boye, 1994). DnaA is found in a wide spectrum of bacteria, and is probably essential in all true bacteria (Skarstad and Boye, 1994). The protein recognizes the origin of replication and binds to a small, specific region of DNA (9 base pair DnaA box). DnaA boxes are frequent around the origin, and recruit the replication machinery to that site (Messer and Weigel, 1997). The sequence of DnaA boxes is very well conserved throughout bacteria: 5'-TT$^{A}/_{T}$ TNCACA-3'[1], where T, C, and A represent three out of four monomeric subunits of the DNA polymer and N represents any of the four (Messer and Weigel, 1997).

A general pattern exists in many bacterial origins of replication (Figure 1): an AT-rich region is usually found upstream from replication initiation proteins binding sites (Marczynski and Shapiro, 1993). In addition to DnaA boxes, found in the origins of all currently characterized bacteria, origins of enteric bacteria also contain DNA adenine methylase (dam) methylation sites (GATC). The replication origin in *E. coli* (Oka,

[1] This expression is similar to regular expressions. A slash means an "OR", 5' and 3' signify DNA directionality.

Sugimoto, and Takanami, 1980) contains four DnaA boxes and twelve dam methylation sites. Dam methylation is found in enteric bacteria (e. g. *E. coli*) and cyanobacteria (Katayama et al., 1997). One function of the dam methylase in enteric bacteria is to synchronize DNA replication (Katayama et al., 1997), but its function in cyanobacteria is unknown. In addition, many bacterial origins contain integration host factor (IHF) binding sites.

Figure 1. Architectural elements at bacterial origins of replication

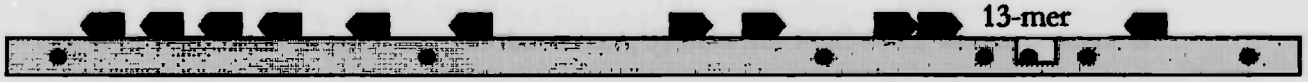
***E. coli* origin (approx. 400 bp)**

DnaA box (blue arrows) consensus: *TTATCCACA*

IHF consensus: $^{A}/_{T}ATCAANNNNTT^{G}/_{A}$

Dam binding sites (filled dots): *GATC*

***Synechococcus* putative origin (270 bp)**

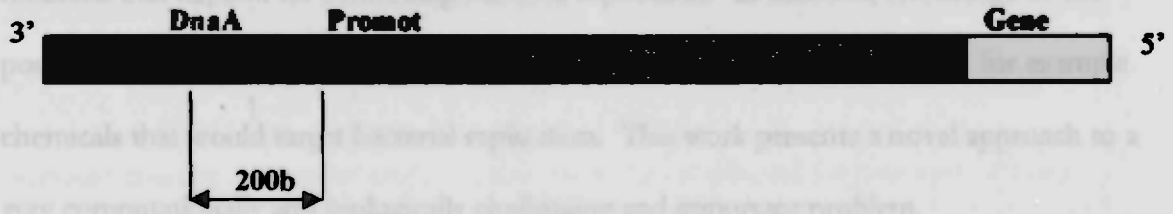


DnaA box consensus: *TTTTCCACA*

Many cellular responses are tied to the cell cycle, so it is significant that DnaA is not only an initiator of DNA replication, but also controls the expression of genes in *E. coli* (Messer and Weigel, 1997). DnaA can act as a repressor or activator of transcription of some genes by binding to DnaA boxes that occur upstream or within the promoter region

of regulated genes (Fig. 2) (Messer and Weigel, 1997). Few genes regulated by DnaA are known, identified haphazardly, and most in *E. coli*. There has been no systematic search for such genes.

Figure 2. A DnaA regulated gene.



Twelve complete bacterial genomes have been sequenced[2], and the origin of replication has been identified in many of the bacteria by biological function (ability to replicate individually), however some still remain unknown. In cyanobacterium *Synechococcus*, a putative origin of replication (Liu et al., 1995), containing eight DnaA boxes and several dam methylation sites, was found computationally. In the replication origin of *Synechococcus*, dam methylase sites occur at a frequency of 7 per 217 base pairs, which is significantly higher than statistical expectation of 1 in 256 base pairs. The location of the origin of replication in another cyanobacterium, *Synechocystis* PCC 6803, whose genome is completely sequenced[3], is unknown. However, it would be reasonable to expect the origin in cyanobacteria to exhibit the same general characteristics as those of enteric bacteria and *Synechococcus*.

[2] http://evolution.bmc.uu.se/~moan/seqdbs.html
[3] http://www.kazusa.or.jp/cyano/cyano.html

Currently, most identification of origins of replication is conducted purely through complex and lengthy biological experiments. The rapid sequencing of microbial genomes presents urgent need for automated methods of genomic analysis. Identification of origins of replication and of genes controlled by DnaA is crucial for development of many research methods that depend on controlling bacterial replication. In addition, knowledge of the position of the origin is important for the development of antibacterial drugs, for example chemicals that would target bacterial replication. This work presents a novel approach to a very computationally and biologically challenging and important problem.

Methods and design

The goal of this study was to develop computer programs to identify the origin of DNA replication in bacterial genomes and to computationally identify genes in *E. coli* whose transcription is regulated by DnaA. Genomes with known origins of replication were used to develop the method. The strategy for identifying origins of replication is being tested on the cyanobacterium *Synechocystis* PCC 6803.

To computationally identify origins of replication, we have developed algorithms based on scoring regions of the genome for number of DnaA sites, AT content, DNA deoxyadenine methylase (Dam) boxes, and integration host factor (IHF) binding sites. DnaA is a protein that regulates DNA replication in bacteria; IHF is a protein that initiates site-specific bending in DNA which facilitates recombination and may also facilitate initiation of DNA replication. Methylation at Dam sites controls synchrony of replication in *E. coli*, and Dam methylation is also universal among cyanobacteria, including *Synechocystis*.

The *Synechocystis* genome was searched for 500-base-pair-long regions containing clusters of DnaA boxes (higher than statistically predicted). The length of the region to consider was elucidated from the sizes of biologically determined origin regions submitted to Genbank[4]. Two DnaA box consensus sequences were used 3' TT$^{T}/_{A}$T N CACA 5' (from determination of binding constants by Schaper and Messer) or a more relaxed consensus 3' $^{T}/_{C}$ $^{T}/_{C}$ T/A/C T N C $^{A}/_{G}$ C/A/T $^{A}/_{C}$ 5' (a general consensus determined from various literature sources). Potential origin regions were then examined for presence of Dam binding sites (*GATC*). The algorithm followed in identification of regions with high number of DnaA binding sites is described in Figure 3 (program in PERL is included in Appendix I).

Figure 3. Algorithm for finding putative origins based on DnaA boxes and Dam methylase sites.

Define: DnaA box
Dam methylase box

Search for DnaA boxes on the genomic sequence, look until found a match. Remember the coordinate of DnaA box as Start of Region.

For 500 bp region after the match:
search for any additional DnaA boxes in the region
search for any Dam binding sites in the region

Print information about the coordinates of the region and of DnaA boxes and Dam methylase sites within the region.

Advance coordinate for Start of Region by 1 (so the next region is displaced one coordinate downstream (to the right)).

[4] http://www.ncbi.nlm.nih.gov/

Two methods were used to determine statistical significance of the regions with DnaA boxes. In the first method, the number of regions in the genome that have N (N = 1, 2, 3, 4) DnaA binding sites was calculated both in *E. coli* and in *Synechocystis* genomes. In addition, the same procedure was repeated for a random sequence with the same base composition and constraints (an inverse of the consensus sequence was used). The procedure was implemented in PERL by modifying algorithm presented in Figure 3 to count the number of regions with N occurrences for $N > 0$, the program is presented in Appendix II. The number of regions with N DnaA binding sites was compared to the number of regions with N occurrences of the random sequence. In addition, the data for *Synechocystis* was compared to that obtained for *E. coli.*

In addition, the expected number of 500 nucleotide long regions with at least one DnaA box was determined to be 109 if DnaA binding site sequence is assumed to occur randomly in the *Synechocystis* genome (Figure 4). It is reasonable to assume that these 109 regions include regions with more than one DnaA box. Because of the independence assumption, poisson distribution can be used to estimate the probability of DnaA boxes being more than 500 bases apart, which was found to be 98.5%. From that, 0.015 of the 109 regions with DnaA binding site are expected to have two DnaA boxes in them and $1.2*10^{-6}$ of the 109 regions would have three DnaA boxes, making expected number of regions be 2 and less than 0, respectively (Figure 4).

Figure 4. Statistical predictions based on random occurrence of DnaA boxes in the genome.

<u>If assume that DnaA boxes occur randomly, then:</u>

$$P(TT{}^{T}\!/_{A}\,TNCACA) = (\frac{1}{4})^{7} * \frac{1}{2} * 1 = 3.05 * 10^{-5}$$

Length of *Synechocystis* genome is 3,573,470 bases

<u>Expected number of regions with at least one DnaA box (overlap allowed):</u>

$$\text{Expected} = P(\text{consensus}) * \text{Length\_of\_genome}$$

$$\text{Expected} = 3.05 * 10^{-5} * 3{,}573{,}470 = 108.9 \approx 109$$

<u>What's the probability of regions with more than one DnaA box?</u>

$$P(\text{dist b/w hits} > 500) = e^{(-p)(500)} = 98.5\%$$

$$P(\text{dist b/w hits} < 500) = 1 - .985 = .015104$$

$$\text{Expected (2boxes)} = .015 * 109 = 1.6 \approx 2$$

$$P(3 \text{ boxes in} < 500) = \text{Gamma}\frac{1}{(p, 3)}$$

$$P(3 \text{ boxes}) = \int_{0}^{500} \frac{1}{(3.05 * 10^{-5})^{3}} x^{2} e^{\frac{-x}{3.05*10^{-5}}} = 1.2 * 10^{-6}$$

$$\text{Expected (3boxes)} = 1.2 * 10^{-6} * 109 = 1.3 * 10^{-4} \approx 0$$

The regions with more than one DnaA box were also examined for the presence of IHF sites (3' $^{A}\!/_{T}$ATCAANNNNTT$^{G}\!/_{A}$ 5' or 3' GNT$^{T}\!/_{C}{}^{A}\!/_{G}$A$^{A}\!/_{T}{}^{A}\!/_{T}{}^{A}\!/_{T}T^{T}\!/_{C}{}^{A}\!/_{G}$ANC 5'). The software used for searching for regions with high DnaA box content was used (Figure 3, Appendix II) with modification that after such regions were found, they were searched for IHF sites instead of Dam methylase binding sites.

In addition, the *Synechocystis* genome was examined for regions with AT content significantly higher than that of the genome. In this algorithm, described in Figure 5, 100 base pair long regions were examined (refer to Appendix III for the program printout). The ratio of A and T nucleotides in the region to the total region length was compared to

the AT content of the genome. In addition, a program was developed to find AT content of a specific region (refer to Appendix IV for the program printout).

Figure 5. Algorithm for finding regions with high AT content (in comparison with the AT content of the genome).

Define: Cutoff

Start examining 100 bp region with start coordinate at Start of Region

calculate number of A and T nucleotides in the region
AT content = (number of A and T)/(length of region)
Ratio = (AT content of region)/(AT content of genome)
if Ratio > Cutoff, then print out region coordinates

Advance coordinate for Start of Region by 10 (so the next region is displaced by 10 nucleotides downstream (to the right)).

Genes in *E. coli* whose transcription could be regulated by DnaA were identified by searching for DnaA boxes proceeding biologically identified promoters (Figs 4, 5). A list of promoters was obtained from Fred Blattner's lab (306 promoters). Positions of DnaA boxes in the genome were compared to the locations of the promoters, and promoters with a DnaA box within 200 base pairs upstream from them were identified.

A suite of three programs were developed to process promoter coordinates file, to locate DnaA sites that have open reading frames within 200 base pairs downstream from then, and to match these DnaA sites with promoters. Program DnaA_prot.pl first searches for DnaA binding sites in *Synechocystis* genome (Figure 6, Appendix V). Once a DnaA box is located, the program scans downstream for open reading frames (orf) and if there is an orf

within 500 bases downstream[5], the coordinates of both DnaA binding site and the longest possible orf are printed out. Program promoters.pl is utilized to create a file with coordinates of biologically identified promoters and information about them extracted from the file with promoter information received from Frad Blattner's laboratory (University of Wiscouncin at Madison) (Appendix VI). Another program, DnaA_promoterOfrs.pl, examines each coordinate from the file created by DnaA_prot.pl and searches the file with promoter information created by promoters.pl for coordinates of promoters within 200 nucleotides downstream of DnaA boxes (Appendix VII).

Figure 6. Algorithm for identification of genes regulated by DnaA.

```
Search the genome for DnaA box
        When a DnaA box is found, scan for a start codon
                if start codon found, search downstream for stop codon
                        if stop codon found, check if it's in frame
                        if in frame, consider this an orf and print coords into File
        Look for next DnaA box, repeat until whole genome searched

Take a DnaA coordinate from the File
        if there is a promoter downstream of that DnaA box (search promoter
file)
        (examine both strands)
        then print out DnaA box and promoter coordinate information
```

[5] Promoters are usually located upstream from the start codon. It is reasonable to assume that a promoter can be anywhere within 300 bases of the start codon. Since DnaA is usually located within 200 bases upstream from DnaA-regulated genes, it is reasonable to consider DnaA boxes up to 500 nucleotides upstream from the start of the orf.

Results and Discussion

Several potential origins of replication have been identified in *Synechocystis* using the relaxed DnaA box sequence, but no region had the number of characteristic sites typical of bacterial origins. Both in E. coli and in Synechocystis, total number of regions with N DnaA boxes was much higher than the total number of regions with N occurences of the randomly generated sequence of the same base composition as a DnaA box (Table 1, Figure 7). The difference is more pronounced in *Synechocystis* than in *E. coli*: the first has 2.3 times more regions which contain one DnaA box than regions with randomly-generated sequence, whereas the fraction is 1.4 in *E. coli*.

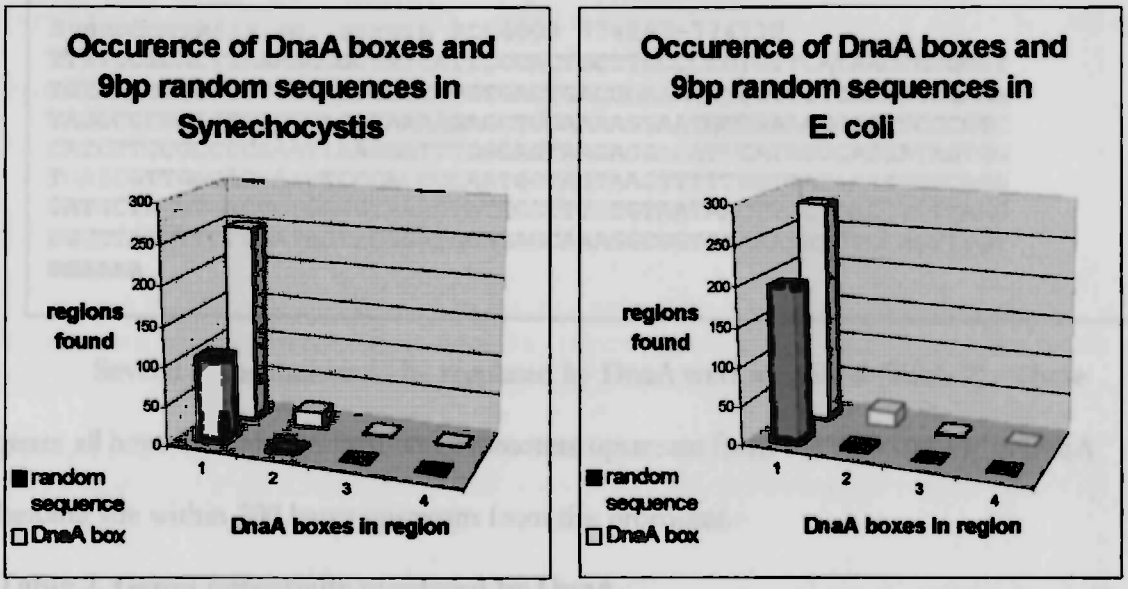
The number of regions with one DnaA box found in Synechocystis with strict DnaA consensus was 250, which greatly exceeds the 109 regions with one or more DnaA box expected in case of random DnaA box distribution (Table 1, Figure 4). The number of regions in which two DnaA boxes were found is 20, which is much larger than the predicted value of 2 in case of random distribution (Table 1, Figure 4). No 500-nucleotide-long regions with three DnaA boxes were expected, however, one was located (Table 1, Figure 4).

Table 1. Comparison of number of DnaA boxes in genome

Occurrences[1]	E.coli		Synechocystis			
	Strict consensus		Strict consensus		Relaxed consensus	
	Random[2]	DnaA box	Random[2]	DnaA box	Random[2]	DnaA box
1	199	283	108	250	2589	1832
2	5	20	0	20	2539	3879
3	0	2	0	1	1311	4032
4	0	1	0	0	429	2738

[1]Number of DnaA boxes in 500bp region
[2]Using DnaA consensus sequenced reversed

Figure 7. Occurence of DnaA boxes and random sequences in genomes of *Synechocystis* and *E. coli*.



When the strict definition of DnaA box was used, a single 500 base pairs long region, which contains three DnaA binding sites and is located at 774287 in the genome was identified (Figure 8). In addition to three DnaA boxes (by strict consensus), this region contains several unidentified repeats, which could be a characteristic of a replication origin. The region does overlap with an open reading frame, whereas *E. coli* origin does not. However, it has not been determined that bacterial origins necessarily have to occur in the intragenic regions.

Although the putative origin region has fewer DnaA binding sites than many bacterial origins, the number of DnaA boxes is still much higher than statistically predicted (Figure 4). In addition, no regions were identified that contained three (or two) occurrences of the randomly generated sequence of the same base composition as DnaA box, indicating that clustering of DnaA boxes in the region of interest is probably not random (Table 1).

Figure 8. Proposed origin of replication identified by using strict DnaA consensus

DnaA boxes are highlighted in blue, green and yellow regions are unidentified repeats.

Synechocystis sp. strain PCC6803 774287-774712

TTTTCCACACTTAAAGCCGCTATCATTCCCACTGCTTGCCCTGTGTTCATAACTAGGGGT
TGTAAACGGGTGCTGCCATTGGCCATGTGACTGACGGAAATATT**TTTTTCACA**GGGTAAA
TAGCCCCGCGCTGTCGGAGATAAAAGAGCTGGAAAAGGAATGGTAAAGGGGGTTCCCGTC
CATCTTCCGCCCCAAATTAAGGATTTGGGAGTAAGAGGAAATTCATAGCCAGGATAGTGG
TGATCGTTGGCATAATTTCCCACCCCAATGCTAGTAACTTTTTGGTTATAAATTGGCAGG
GATGCTACCTGACCCTGGGGTAAAATATCCCTTTCTGTAATAACTTGCTGACCTTTTAAG
CGCCTACTTTCCCGATAGTAGGGATGGAGAGCAAAGGCGGTGGAAATATCTCCAGTT**TGT**
GGAAAA

Several genes that could be regulated by DnaA were identified (Table 2). These genes all have biologically identified promoters upstream from the start site and a DnaA binding site within 200 bases upstream from the promoter.

Table 2. Genes potentially regulated by DnaA

Gene name	Distance between promoter and DnaA box*	Function
queA	12	tRNA modification
sdh	169 from sdhp2	Hydrogenase
putP	68 from putPp5	proline utilization
purR	50	Regulatory gene for pur regulon
pfkB	85	level of 6-phosphofructokinase
fadL	14	fatty acids transport
gut	47	gut operon – dehydrogenase, phosphotransferase
rpmH	132 from rpmHp1	50S ribosomal subunit protein
ilvG	181	Acetolactate synthase II
rho	117	Transcription termination factor
btuB	154	receptor for vitamin B12
lexA	46	Regulatory gene for SOS operon
argF	52	ornithine carbomoyltransferase (cytochrome B)
fabA	54	beta-Hydroxydecanoyl thioester dehydrase
hisA	12	Histidine
dapA	12	Dihydrodipicolinate synthase
ansB	76	Asparaginase
trmA	94	tRNA for a rare nucleotide

*DnaA box within 200bp of promoter identified

The genes identified have diverse functions and no definite link between their function and initiation of replication can be established for most of them. Some of the genes have multiple promoters, which implies that DnaA protein could be regulating one or several promoters by its binding.

Literature Cited

Liu, Y., and N. Tsinoremas (1995). An unusual gene arrangement for the putative chromosome replication origin and circadian expression of *dnaN* in *Synechococcus* sp. Strain PCC 7942. *Gene* 172:105-109.

Katayama, T., N. Akimitsu, T. Mizushima, T. Miki, K. Sekimizu (1997). Overinitiation of chromosome replication in *Escherichia coli dnaAcos* mutant depends on activation of oriC function by the *dam* gene product. *Mol. Microbiol.* 25:661-670.

Marczynski, G., and L. Shapiro (1993). Bacterial chromosome origins of replication. *Curr. Opin. Genet. and Develop.* 3:775-782.

Messer, W., and C. Weigel (1997). DnaA initiator - also a transcription factor. *Molecular Microbiology* 24:1-6.

Oka, A., Sugimoto, K., and M. Takanami (1980). Replication origin of the Escherichia coli K-12 chromosome: the size and structure of the minimum DNA segment carrying the information for autonomous replication. *Molec. Gen. Genet.* 178:9-20.

Richter, S. and W. Messer (1995). Genetic structure of the *dnaA* region of cyanobacterium Synechocystis sp. strain PCC6803. *J. Bacteriol.* 177:4245-4251.

Skarstad, K., and E. Boye (1994). The initiator protein DnaA: evolution, properties and function. *Biochim. Biophys. Acta.* 1217:111-130.

Appendix I

```
#!/usr/bin/perl

#####################################################################
#Author: Olga Troyanskaya
#File: origins_Dam.pl
#Description: program searches for DnaA binding sites, and finds
#          Dam methylase binding sites within identified regions.
#Date last modified:
#####################################################################
#To run the program: origins_Dam.pl <input_file> <output_file>
#                where <input_file> has genename on first line
#                and sequence with no returns on next line
#                (fasta2seq.pl can be used for seq. conversion).
#
# if no output file is specified, output will appear in the ter-
# minal window (standard output will be used)
#####################################################################

#open necessary files
$infile = $ARGV[0]; #name of input file is the first parameter

if (@ARGV eq 2) {  #open output file
   $outfile = $ARGV[1];  #output file is the second parameter
   unless (open(OUTFILE, ">$outfile")) {
      die ("Cannot open output file $outfile\n");
   }
   select (OUTFILE);
}
else {
   select (STDOUT); #if no output file specified, use STDOUT
}

#define DnaA box pattern and its reverse complement
$DnaAbox = "(TT[TA]T[CAGT]CACA)|(TGTG[ACTG]A[TA]AA)"; #E. coli consensus from
determination of binding constants (Schaper and Messer)
$damBox = "GATC";

#initialize count vars
$damCount = 0; #keeps track of number of dam binding sites found
$count = 0; #var for array index
$index = 0; #count of the number of 2000bp regions with high frequency of DnaA boxes

unless (open(INFILE, $infile)) {   #open input file
   die ("Cannot open input file $infile\n");
}

$genename = <INFILE>; #in fasta format, the first line is info
                #on the gene

$line = <INFILE>;

#find all occurences of pattern in $line.  index will find next position of
#its second argument in its first argument (if third argument is given, the
#number of chars given in the third argument is skipped)
```

```
while ($line =~ /$DnaAbox/ig) {

   @region[$index] = $count; #store the array index of the first match in the region
   @begin[$count] = pos($line) +1 - length($&); #the beginning of the match
   @end[$count] = @begin[$count] + length($&) -1 ; #the end of the match
   $DnaAmatch[$count] = $&; #store the match
   @beginRegion[$index] = @begin[$count]; #start coord of 2000 bp region
         $count++;


   #now search the 2000 bp region after the match for any other DnaA boxes
   $nextpos = @begin[$count-1] + 1;  #position to start 2000 bp region
   $offset = $nextpos; #store the start coordinate to figure out coords later

   $region = substr($line, $nextpos, 500); #create a substring for the
                                 #next 2000nb (change this number
                                 #to change the length of the regions)

   while ($region =~ /$DnaAbox/ig) {
         #while still in the same 2000bp region, find more matches
     #record info in arrays
         @begin[$count] = $offset + pos($region) + 1 - length($&); #the beginning of the match
         @end[$count] = $begin[$count] + length($&) - 1; #the end of the match
         $DnaAmatch[$count] = $&; #store the match
         #print ("count $count begin $begin[$count]\n");
         $count++;
   }

   #once finished parcing the 2000 bp region
   if (($count-$region[$index])<3) { #if region has less than n dnaA boxes
         #then forget it
         $count = @region[$index]; #reset count back to the index of the first match in the region

   } else {
         #if 3 or more DnaA boxes in the region, then consider region valid
         @endRegion[$index] = $end[$count-1]; #end coord of the region
         #use count-1 b/c count has
         #already been advanced

         @damIndex[$index] = $damCount; #store the index to first dam site in the region
         while ($region =~ /$damBox/ig) {
            #find dam methylase sites
            @beginDam[$damCount] = $offset + pos($region) + 1 - length($&); #the beginning of the
                                                                              match
            @endDam[$damCount] = $offset + pos($region); #the end of the match
            #print ("dam match $damCount at $beginDam[$damCount]\n");
            $DamMatch[$damCount] = $&; #store the match
            $damCount++;

   }

         $index++;
   }
}
```

```
#printing results
$region[$index] = $count;
$damIndex[$index] = $damCount;

for ($i = 0; $i <= ($index-1); $i++) {

   print ("\n\n****region $i from @beginRegion[$i] to @endRegion[$i]****\n");
   $c = $region[$i];·                    #index of the first match

   for ( ; $c < $region[$i+1]; $c++) { #print all the matches within the region
         print ("DnaA match $begin[$c] to $end[$c]  $DnaAmatch[$c]\n");
   }

   $lcv = $damIndex[$i];                 #index of the first dam match
   $numberDamMatches = 0; #count number of Dam matches in the region
   for ( ; $lcv < $damIndex[$i+1]; $lcv++) { #print all the matches within the region
         print ("Dam match $beginDam[$lcv] to $endDam[$lcv]   $DamMatch[$lcv]\n");
         $numberDamMatches++;
   }
 print ("$numberDamMatches Dam binding sites found\n");

}
```

Appendix II

```
#!/usr/bin/perl

#####################################################################
#Author: Olga Troyanskaya
#File: origins_IHF.pl
#Description: program searches for DnaA binding sites, and finds
#            IHF binding sites in the region
#Date last modified: 9/15/98
#####################################################################
#To run the program: origins_IHF.pl <input_file> <output_file>
#                    where <input_file> has genename on first line
#                    and sequence with no returns on next line
#                    (fasta2seq.pl can be used for seq. conversion).
#
# if no output file is specified, output will appear in the ter-
# minal window (standard output will be used)
#####################################################################

#open necessary files
$infile = $ARGV[0]; #name of input file is the first parameter
print ("Hello\n");

if (@ARGV eq 2) {  #open output file
   $outfile = $ARGV[1];  #output file is the second parameter
   unless (open(OUTFILE, ">$outfile")) {
      die ("Cannot open output file $outfile\n");
   }
  select (OUTFILE);
}
else {
   select (STDOUT); #if no output file specified, use STDOUT
}

$IHFBinSite = "([TC][Tc][TAc]T[CAGT]C[Ag][CAt][Ac])|([Tg][TGa][Tc]G[TGAC]A[gTA][Ag][AG])";

$DnaAbox = "(AGATCT[ATGC]TTTATT|AGATCTGTT[ATGC]TAT|TGATCTCTTATTAGG)"; #from
Salzberg et al.

#initialize count vars
$IHFcount = 0; #keeps track of number of IHF binding sites found
$count = 0; #var for array index
$index = 0; #count of the number of 2000bp regions with high frequency of DnaA boxes

unless (open(INFILE, $infile)) {   #open input file
   die ("Cannot open input file $infile\n");
}

$genename = <INFILE>; #in fasta format, the first line is info
                #on the gene
print ("hi");
$line = <INFILE>;
print ("got line");
#find all occurences of pattern in $line.  index will find next position of
#its second argument in its first argument (if third argument is given, the
#number of chars given in the third argument is skipped)
```

```
study($line);

while ($line =~ /$DnaAbox/ig) {

   $region[$index] = $count; #store the array index of the first match in the region
   $begin[$count] = pos($line) +1 - length($&); #the beginning of the match
   $end[$count] = $begin[$count] + length($&) -1 ; #the end of the match
   $DnaAmatch[$count] = $&; #store the match
   $beginRegion[$index] = $begin[$count]; #start coord of 2000 bp region
   $count++;


   #now search the 2000 bp region after the match for any other DnaA boxes
   $nextpos = $begin[$count-1] + 1;  #position to start 2000 bp region
   $offset = $nextpos; #store the start coordinate to figure out coords later

   $region = substr($line, $nextpos, 500); #create a substring for the
                                 #next 2000nb (change this number
                                 #to change the length of the regions)

   while ($region =~ /$DnaAbox/ig) {
         #while still in the same 2000bp region, find more matches
     #record info in arrays
         $begin[$count] = $offset + pos($region) + 1 - length($&); #the beginning of the match
         $end[$count] = $begin[$count] + length($&) - 1; #the end of the match
         $DnaAmatch[$count] = $&; #store the match
         #print ("count $count begin $begin[$count]\n");
         $count++;
   }



   #once finished parcing the 2000 bp region
   if (($count-$region[$index])<2) { #if region has less than n dnaA boxes
         #then forget it
         $count = $region[$index]; #reset count back to the index of the first match in the region

   } else {
         #if 3 or more DnaA boxes in the region, then consider region valid
         $endRegion[$index] = $end[$count-1]; #end coord of the region
         #use count-1 b/c count has
         #already been advanced

         $IHFIndex[$index] = $IHFcount; #store the index to first IHF site in the region
         while ($region =~ /$IHFBinSite/ig) {
            #find IHF sites
            $beginIHF[$IHFcount] = $offset + pos($region) + 1 - length($&); #the beginning of the match
            $endIHF[$IHFcount] = $offset + pos($region); #the end of the match
            #print ("IHF match $IHFcount at $beginIHF[$IHFcount]\n");
            $IHFMatch[$IHFcount] = $&; #store the match
            $IHFcount++;

   }

         $index++;
```

```
    }
}


for ($i = 0; $i <= ($index-1); $i++) {

    print OUTFILE ("\n\n****region $i from $beginRegion[$i] to $endRegion[$i]****\n");
    $c = $region[$i];                    #index of the first match

    for ( ; $c < $region[$i+1]; $c++) { #print all the matches within the region
            print OUTFILE ("DnaA match $begin[$c] to $end[$c]  $DnaAmatch[$c]\n");
    }

    $lcv = $IHFIndex[$i];                #index of the first IHF match
    $numberIHFMatches = 0; #count number of IHF matches in the region
    for ( ; $lcv < $IHFIndex[$i+1]; $lcv++) { #print all the matches within the region
            print OUTFILE ("IHF match $beginIHF[$lcv] to $endIHF[$lcv]   $IHFMatch[$lcv]\n");
            $numberIHFMatches++;
    }
    print OUTFILE ("$numberIHFMatches IHF binding sites found\n");

}
```

Appendix III

```
#!/usr/bin/perl

#####################################################################
#Author: Olga Troyanskaya
#File: AT_top.pl
#Description: Relative likelihood should do:  compute P(AT) for
#                       the whole genome, and then measure P(A) and P(T)
#               within the region of interest (call this P(ATr)).
#               Then compute P(ATr)/P(AT) which should be > 1
#               if the region is AT-rich.
#
#Date last modified:
#####################################################################
#To run the program: AT.pl <input_file> <output_file> cutoff
#               where <input_file> has genename on first line
#               and sequence with no returns on next line
#               (fasta2seq.pl can be used for seq. conversion).
#
#####################################################################


#open necessary files
$infile = $ARGV[0]; #name of input file is the first parameter
$outfile = $ARGV[1];  #output file is the second parameter
$cutoff = $ARGV[2]; #cutoff for which regions AT content to print out

unless (open(INFILE, $infile)) {   #open input file
   die ("Cannot open input file $infile\n");
}
unless (open(OUTFILE, ">$outfile")) { #open output file
   die ("Cannot open output file $outfile\n");
}

$genename = <INFILE>; #in fasta format, the first line is info
                #on the gene

$line = <INFILE>;
$targetString = "[AT]";
$regionLength = 100;
$startRegion = 0;
#size of E. coli genome is 4639221
#size of Synechocystis PCC 6803 genome is 3573470
$genomeSize = 3573470;
study($line);

#calculate P(AT) for the genome
$ATcounter = 0;
while ($line =~ /$targetString/ig) {
   $ATcounter++;
}

chop($line);
$PatGenome = $ATcounter/length($line);
```

```
while ($startRegion < $genomeSize) {
   $region = substr($line, $startRegion, $regionLength);

   $ATcounter = 0;
   while ($region =~ /$targetString/ig) {
         $ATcounter++;
   }

   $PatRegion = $ATcounter/$regionLength; #proportion of AT in region


   $ratio = $PatRegion/$PatGenome;
   if ($ratio > $cutoff) {
         $endRegion = $startRegion + $regionLength;
         print OUTFILE ("***Region from $startRegion to $endRegion\n");
         print OUTFILE ("Genome P(AT) is $PatGenome, region P(AT) is $PatRegion\n");
         print OUTFILE ("Ratio is $ratio\n\n");
   }

   $startRegion = $startRegion + 10;
}
```

Appendix IV

```
#!/usr/bin/perl

#####################################################################
#Author: Olga Troyanskaya
#File: AT.pl
#Description: Relative likelihood should do:  compute P(AT) for
#       the whole genome, and then measure P(A) and P(T)
#       within the region of interest (call this P(ATr)).
#       Then compute P(ATr)/P(AT) which should be > 1
#       if the region is AT-rich.
#
#Date last modified:
#####################################################################
#To run the program: AT.pl <input_file> <output_file> <regionst> <regionend>
#           where <input_file> has genename on first line
#           and sequence with no returns on next line
#           (fasta2seq.pl can be used for seq. conversion).
#
#####################################################################


#open necessary files
$infile = $ARGV[0]; #name of input file is the first parameter
$outfile = $ARGV[1];  #output file is the second parameter
$startRegion = $ARGV[2];
$endRegion = $ARGV[3];

unless (open(INFILE, $infile)) {   #open input file
   die ("Cannot open input file $infile\n");
}
unless (open(OUTFILE, ">$outfile")) { #open output file
   die ("Cannot open output file $outfile\n");
}

$genename = <INFILE>; #in fasta format, the first line is info
            #on the gene

$line = <INFILE>;
$targetString = "[AT]";
$regionLength = $endRegion - $startRegion + 1;


$region = substr($line, $startRegion, $regionLength);


while ($region =~ /$targetString/ig) {
   $ATcounter++;
}

$PatRegion = $ATcounter/$regionLength; #proportion of AT in region

$ATcounter = 0;

while ($line =~ /$targetString/ig) {
   $ATcounter++;
```

```
}

chop($line);
$PatGenome = $ATcounter/length($line);

$ratio = $PatRegion/$PatGenome;
print OUTFILE ("Genome P(AT) is $PatGenome, region P(AT) is $PatRegion\n");
print OUTFILE ("Ratio is $ratio\n");
```

Appendix V

```
#!/usr/bin/perl

#####################################################################
#Author: Olga Troyanskaya
#File: DnaA_prot.pl
#Description: program searches for DnaA binding sites, then
#       looks for orfs within specified distance downstream
#       of DnaA boxes
#Date last modified:
#####################################################################
#To run the program: DnaA_prot.pl <input_file> <output_file>
#           where <input_file> has genename on first line
#           and sequence with no returns on next line
#           (fasta2seq.pl can be used for seq. conversion).
#
# if no output file is specified, output will appear in the ter-
# minal window (standard output will be used)
#####################################################################

#some constants
$minOrfLength = 240;
$maxRegionLength = 500; #max length of region b/w DnaA boxes and ATG

#open necessary files
$infile = $ARGV[0]; #name of input file is the first parameter


if (@ARGV eq 2) {  #open output file
   $outfile = $ARGV[1];  #output file is the second parameter
   unless (open(OUTFILE, ">$outfile")) {
      die ("Cannot open output file $outfile\n");
   }
   select (OUTFILE);
}
else {
   select (STDOUT); #if no output file specified, use STDOUT
}

$DnaAbox = "(TT[TA]T[CAGT]CACA)|(TGTG[ACTG]A[TA]AA)"; #E. coli consensus from
determination of binding constants (Schaper and Messer)

#initialize some vars
$count = 0; #var for array index
$index = 0; #count of the number of 2000bp regions with high frequency of DnaA boxes
$foundATG = 0;
$stopInFrame = 0;
$orfFound = 0;

unless (open(INFILE, $infile)) {   #open input file
   die ("Cannot open input file $infile\n");
}

$genename = <INFILE>; #in fasta format, the first line is info
```

```
            #on the gene

$line = <INFILE>;

#find all occurences of pattern in $line. index will find next position of
#its second argument in its first argument (if third argument is given, the
#number of chars given in the third argument is skipped)


while ($line =~ /$DnaAbox/ig) {

   $region[$index] = $count; #store the array index of the first match in the region
   $begin[$count] = pos($line) +1 - length($&); #the beginning of the match
   $end[$count] = $begin[$count] + length($&) -1 ; #the end of the match
   $DnaAmatch[$count] = $&; #store the match
   $beginRegion[$index] = $begin[$count]; #start coord of 2000 bp region

   #now search the region after the match for orfs
   $nextpos = $end[$count];
   $offset = $nextpos; #store the start coordinate to figure out coords later

   $region = substr($line, $nextpos, $maxRegionLength); #create a substring for the
                              #next 200nb (change this number
                              #to change the length of the regions)
   #scan region for ATG or GTG (start) codon
   while (($region =~ /ATG|GTG/ig) && (!$orfFound)) {
         $startPosition = $offset + pos($region) +1 - length($&);
         $npos = pos($region) + 1; #position to start next pattern search for ATG from
         $foundATG = 1;
         #print ("start found at $startPosition\n");



   if ($foundATG == 1) {
         $orfRegion = substr($line, $startPosition);

         while (($orfRegion =~ /(TAG)|(TGA)|(TAA)/ig)&&(!$stopInFrame)) {
         #scan for stop codons
            $lengthOrfRegion = (pos($orfRegion) + 1); #since start position is
                                    #the start of orfRegion

            if (!($lengthOrfRegion%3)) { #if length of region divisible by 3
                                 #then the stop codon is in frame
                           #x%y == 0 iff x is divisible by y
               $stopInFrame = 1;  #stop codon in frame found
               if ($lengthOrfRegion > $minOrfLength) {
                  $orfFound = 1;
                  $beginOrf[$count] = $startPosition;
                  $endOrf[$count] = $startPosition + pos($orfRegion); #the end of the match
                  $orfMatch[$count] = $&; #store the match
               }
            }
         }
   }
         pos($region) = $npos; #examine next plausible start
         $stopInFrame = 0;
```

```
    }

    $orfFound = 1; #take this out later
        if ($orfFound) {
            #consider region valid
            $endRegion[$index] = $end[$count-1]; #end coord of the region
            #use count-1 b/c count has
            #already been advanced
            pos($line) = $nextpos + 1; #have to tell pattern matching where to start
            $posit = pos($line);
            #print ("new position $posit\n");
            $index++;
            $count++;
            $orfFound = 0;
            $foundATG = 0;
        }
    }


#printing results
$region[$index] = $count;

for ($i = 0; $i <= ($index-1); $i++) {

  # print ("\n\n****region $i from $beginRegion[$i] to $endRegion[$i]****\n");
   $c = $region[$i];                    #index of the first match

   for ( ; $c < $region[$i+1]; $c++) { #print all the matches within the region
        print ("match $c\n");
        print ("DnaA match $begin[$c] to $end[$c]  $DnaAmatch[$c]\n");
        $dist = $beginOrf[$c] - $begin[$c];
        print ("orf from $beginOrf[$c] to $endOrf[$c] distance = $dist\n");

        #print ("$orfMatchp[$c]\n");
        print ("\n");
    }
}
```

Appendix VI

```
#!/usr/bin/perl

##########################################################################
##########
#Author: Olga Troyanskaya
#File: promoters.pl
#Description: extracts promoter information from the file from Wis_Mad with
#           annotated E. coli data
#To run: promoters.pl <input file> <word to look for>
##########################################################################
##########

$infile = $ARGV[0];
$query = "$ARGV[1]"; #word to look for in a file

unless (open(INFILE, "$infile")) {
   die ("Cannot open input file $infile");
}

$done = 0;
$line = <INFILE>;
while (!eof) {
   if (!($line eq "\n")) { #skip empty lines
         chop($line);  #removes the eol character
         @arr = split(/[\t ]+/, $line); #split line into words
         $seqType = @arr[1];
         if ($seqType =~ /$query/) { #if found the word we are looking for (case sensitive search, to ignore
case add i before the closing bracket (/$query/i))
            print ("$line\n");
            while (!$done) {
                  $line = <INFILE>;
                  chop($line);
                  @arr = split(/[\=\t ]+/, $line); #split line into words
                  if ($arr[1] =~ /\/note/) { #$arr[0] is for some reason a space
                     $note = 1;
                  }
                  if ($note == 1) { #note can be more than 1 line long
                     @arr = split(/[\t ]*/, $line); #split line into chars
                     $l = @arr; #length of array
                     if ($arr[$l-1] =~ /\"/) {     #once got to the second "
                           $done = 1;
                     }
                  }
                  print ("$line\n");
                  if ($done) {
                     print ("*\n"); #just a separator character
                     $note = 0;
                  }
            }
         }
   }
   $done = 0;
   $line = <INFILE>;
}
```

Appendix VII

```
#!/usr/bin/perl

#########################################################################
#Author: Olga Troyanskaya
#File: DnaA_promoterOrfs.pl
#Description: this works on ecoli_promoters.data (file from Wis_Mad with
#           annotated E. coli data
#To run: promoters.pl <input file> <word to look for>
#########################################################################

$strand =1; #template strand by default

$DnaAfile = $ARGV[0];
$promoterFile = $ARGV[1];
$strand = "$ARGV[2]"; #1 for template, 2 for complementary
                #with c strand matches
unless (open(INFILE, "$DnaAfile")) {
  die ("Cannot open input file $DnaAfile");
}

unless (open(PFILE, "$promoterFile")) {
  die ("Cannot open input file $promoterFile");
}

$counter = 0;
$done = 0;
$line = <INFILE>;
while (!eof) {
  if (!($line eq "\n")) { #skip empty lines
        chop($line);  #removes the eol character
        @arr = split(/[\t ]+/, $line); #split line into words

        if (@arr[0] =~ /DnaA/) { #this line contains DnaA box coords
          seek(PFILE, 0, 0); #skip back to the beginning of the promoter file

          $endDnaA = @arr[4]; #end coord for DnaA

          while (!$done && ($pline = <PFILE>)) {
                #start going through the promoter file
                @arr = split(/[\.\(\)\t ]+/, $pline); #split line into words (this line contains promoter
coordinates

                if (@arr[1] =~ /promoter/) {

                  if ($arr[2] =~ /complement/) { #promoter on the complement strand

                        if ($strand == 1) { #template
                          while (!($pline eq "*\n")) {
                                $pline = <PFILE>; #skip to the next promoter site
                          }
```

```
            } else {  #looking at complementary strand

              $pstart = @arr[3];

              if (($endDnaA < ($pstart-5)) && ($endDnaA >= ($pstart - 100))) {
                    print ("$line\n");
                    #print ("$pline");
                    $d = $pstart - $endDnaA;
                    print ("\t\t promoter is $d from DnaA box\n");
                    $done = 1;
                    $counter++; #count the number of hits
              }
            do {
                    if ($done) {
                      print ("$pline");
                    }
                    $pline = <PFILE>; #skip to the next promoter site
              } until ($pline eq "*\n");

              if ($done) {
                    print ("\n");
              }
            }
      } else { #promoter on template strand
            if ($strand == 2) {
              while (!($pline eq "*\n")) {
                    $pline = <PFILE>; #skip to the next promoter site
              }
            } else {  #looking at template strand
              $pstart = @arr[2];

              if (($endDnaA < ($pstart-10)) && ($endDnaA >= ($pstart - 200))) {
                    print ("$line\n");
                    $l = $pstart - $endDnaA;
                    print ("\t promoter is $l from DnaA box\n");
                    $done = 1;
                    $counter++; #count the number of hits
              }
              do {
                    if ($done) {
                      print ("$pline");
                    }
                    $pline = <PFILE>; #skip to the next promoter site
              } until ($pline eq "*\n");

              if ($done) {
                    print ("\n");
              }
            }
      }
    }
}
$done = 0;
```

```
        }
    }
    $line = <INFILE>; #read in next line from DnaA file
}

print ("Total number of hits is $counter\n");
```