Analysis of upstream region of antibiotic resistance genes to identify leader peptides

BY

SAI LAKSHMI SUBRAMANIAN
Bachelor of Engineering, Electronics and Communication, University of Madras, 2003
Master of Science, Bioinformatics, University of Madras, 2005

THESIS

Submitted as partial fulfillment of requirements for the degree of
Master of Science in Bioinformatics in the
Graduate College of the University of Illinois at Chicago, 2011

Chicago, Illinois

Defense Committee:

Alexander Mankin, Chair and Advisor, Center for Pharmaceutical Biotechnology
Jie Liang
Yang Dai
ACKNOWLEDGEMENTS

I am extremely thankful to my advisor Dr. Alexander (Shura) Mankin, for being a true source of inspiration for my work. His guidance, constant encouragement and support have been very helpful throughout the year I worked with him on this project. Dr. Mankin is one of the best mentors I have had, who has taken genuine interest in my life and I sincerely owe this degree to him. I extend my heartfelt thanks to my other committee members Dr. Yang Dai and Dr. Jie Liang.

I am also grateful to all members of the lab, including Dr. Nora Vazquez-Laslop, Dr. Haripriya Ramu, Dorota Klepacki, Jacqueline LaMarre, Krishna Kannan and Pulkit Gupta, for their support and being such wonderful friends.

I would like to express my love and sincere gratitude to my dearest parents Subramanian and Jayshree Subramanian and husband Swaminathan, for their love, patience and constant encouragement in making this happen and making me a better person that I am today. Finally, I dedicate this thesis to my little bundle of joy Rahul who was growing inside me when I was working on this project and for being the most understanding and cooperative toddler while preparing this manuscript.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>..........................................................</td>
</tr>
<tr>
<td>A. Ribosome and its role in protein synthesis</td>
<td>..........................................................</td>
</tr>
<tr>
<td>B. Regulation of translation by ribosome stalling</td>
<td>..........................................................</td>
</tr>
<tr>
<td>C. Antibiotics and antibiotic resistance</td>
<td>..........................................................</td>
</tr>
<tr>
<td>D. Programmed drug-dependent ribosome stalling</td>
<td>..........................................................</td>
</tr>
<tr>
<td>E. Regulation of macrolide resistance genes by ribosome stalling</td>
<td>..........................................................</td>
</tr>
<tr>
<td>II. MOTIVATION</td>
<td>..........................................................</td>
</tr>
<tr>
<td>A. Protein Translation</td>
<td>..........................................................</td>
</tr>
<tr>
<td>B. Sequence features involved in translation</td>
<td>..........................................................</td>
</tr>
<tr>
<td>C. Variability of regulatory leader peptides</td>
<td>..........................................................</td>
</tr>
<tr>
<td>III. METHODS AND IMPLEMENTATION</td>
<td>..........................................................</td>
</tr>
<tr>
<td>A. Short ORF finding program</td>
<td>..........................................................</td>
</tr>
<tr>
<td>1. Form Input</td>
<td>..........................................................</td>
</tr>
<tr>
<td>2. Search Strategy</td>
<td>..........................................................</td>
</tr>
<tr>
<td>3. Search Results</td>
<td>..........................................................</td>
</tr>
<tr>
<td>B. Software</td>
<td>..........................................................</td>
</tr>
<tr>
<td>IV. RESULTS AND DISCUSSION</td>
<td>..........................................................</td>
</tr>
<tr>
<td>A. Dataset used in the analysis</td>
<td>..........................................................</td>
</tr>
<tr>
<td>B. Motif groups in leader peptides</td>
<td>..........................................................</td>
</tr>
<tr>
<td>C. Resistance profile and mechanism of antibiotic resistance genes</td>
<td>..........................................................</td>
</tr>
<tr>
<td>D. Identification of putative leader peptides</td>
<td>..........................................................</td>
</tr>
<tr>
<td>V. CONCLUSION</td>
<td>..........................................................</td>
</tr>
<tr>
<td>CITED LITERATURE</td>
<td>..........................................................</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>..........................................................</td>
</tr>
<tr>
<td>VITA</td>
<td>..........................................................</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. SEQUENCES OF KNOWN LEADER PEPTIDES OCCURRING IN THE UPSTREAM REGION OF MACROLIDE RESISTANCE GENES</td>
<td>11</td>
</tr>
<tr>
<td>II. LEADER PEPTIDES OF MLS&lt;sub&gt;B&lt;/sub&gt; RESISTANCE GENES REPORTED IN LITERATURE AND IDENTIFIED BY THE shORF FINDING PROGRAM</td>
<td>23</td>
</tr>
<tr>
<td>III. RESISTANCE PROFILE AND MECHANISM OF ACTION OF ANTIBIOTIC RESISTANCE GENES</td>
<td>27</td>
</tr>
<tr>
<td>IV. SEQUENCE FEATURES OF KNOWN AND PREDICTED LEADER ORFs OF ANTIBIOTIC RESISTANCE GENES</td>
<td>34</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ribosome structure and function..</td>
<td>2</td>
</tr>
<tr>
<td>2. Nascent peptide and macrolide antibiotic in the exit tunnel of the ribosome.</td>
<td>4</td>
</tr>
<tr>
<td>3. <em>ermC</em> cassette with the leader ORF <em>ermCL</em></td>
<td>6</td>
</tr>
<tr>
<td>4. Regulation of expression of <em>ermC</em> by translation attenuation</td>
<td>8</td>
</tr>
<tr>
<td>5. Sequence features in the RBS of a typical bacterial gene</td>
<td>13</td>
</tr>
<tr>
<td>6. Screen-shot of shORF finder search form</td>
<td>19</td>
</tr>
<tr>
<td>7. Search results showing all possible short peptides of length 4-50 amino acids that can be translated from the query nucleotide sequence</td>
<td>20</td>
</tr>
</tbody>
</table>
LIST OF ABBREVIATIONS

bp  Base pair
CGI  Common Gateway Interface
DHTML  Dynamic HTML
HTML  Hyper Text Markup Language
MLS_B  Macrolide, lincosamide and streptogramin B
ORF  Open reading frame
PTC  Peptidyl transferase center
rRNA  Ribosomal RNA
SD  Shine-Dalgarno
shORF  Short open reading frame
tRNA  Transfer RNA
SUMMARY

The main aim of this project was to analyze the upstream regions of antibiotic resistance genes in order to identify putative short open reading frames (ORFs) that are likely to regulate the expression of the resistance genes. The larger goal was to characterize the mechanism of programmed drug- and nascent peptide-dependent ribosome stalling wherein the leader peptides are responsible for the regulation of expression of the resistance genes. This study involved the analysis of genes known to confer resistance to the macrolide, lincosamide and streptogramin B (MLS$_B$) classes of antibiotics. These genes are known to be inducible, so that the resistance gene is turned on only in the presence of the antibiotic.

Previous studies have shown that some of these genes possess short ORFs in their upstream regions encoding so called leader peptides, which are expressed constitutively in the absence of inducing antibiotic, preventing the ribosome to access the resistance gene. However, in the presence of the inducing antibiotic, the ribosome stalls at specific codons in the leader ORFs, leading to activation of translation of the downstream resistance gene. Such a mechanism is defined as programmed drug- and nascent-peptide dependent ribosome stalling. This mechanism was identified first in the well-characterized $ermC$ methyltransferase gene causing resistance to macrolides, lincosamides and streptogramins B (MLS$_B$ resistance) in bacteria. The ribosome stalls at the ninth codon of the $ermCL$ leader ORF in the presence of the inducing macrolide antibiotic. Thus the sequence of the leader peptide along with the structure of the
SUMMARY (continued)

inducing antibiotic are key players in the formation of the stalled ribosome complex and hence regulation of gene expression.

A few other genes causing MLS$_B$ resistance by ribosomal stalling have also been studied but many others that may be regulated in a similar fashion have not been analyzed so far. Thus the best way to understand the mechanism of regulation of resistance genes by ribosome stalling is to start with the analysis of the upstream leader regions to identify potential candidate regulatory ORFs.

A web-based short ORF (shORF) finding program was developed to identify all possible short ORFs in the upstream regions of antibiotic resistance genes. Though there are many ORF finding programs available, this tool was specifically developed to identify short ORFs, considering all the key sequence features pertinent to known leader peptides like alternative start codons (AUG, GUG and UUG), different sub-sequences of the Shine-Dalgarno (SD) sequence 5'-UAAGGAGGU-3', variable spacer regions between the SD sequence and the start codon of the putative leader ORF and the possibility of leader genes having a very weak SD or lacking SD sequence altogether.

The list of all rRNA methylases, efflux and inactivating enzymes known to cause MLS$_B$ resistance in various strains of bacteria were collected and a comprehensive analysis was performed using the shORF finding program to identify potential candidate leader ORFs. The information on inducibility of these genes, the antibiotic causing
induction, resistance profile and the mechanism of regulation of gene expression for known genes was compiled on basis of the an extensive survey of available literature. The study revealed that around 30% of the known resistance genes possess shORFs, some which have been reported earlier while others are novel and have similarities to other known ORFs.

The inducibility of some of these novel ORFs are unknown and the presence of such ORFs with well defined SD sequence and optimal spacer between the RBS and the start codon clearly suggest that these genes may be expressed and thus, regulated by nascent-peptide dependent ribosome stalling. Ribosome stalling at some of the candidate ORFs has been experimentally validated confirming their role in regulation of gene expression. The bioinformatics analysis did not overlook those short peptides, which did not have a SD sequence since there are many leaderless genes in bacteria with very weak or no SD sequence. The leader peptides were aligned to identify common sequence motifs and the short peptides were classified into several groups based on the presence of these motifs. The majority of the sequences fell into the groups with motifs ‘IFVI’, ‘RLR’ and ‘IAVV’.

This study is a prelude to gain deeper insights into the diversity of short leader peptides involved in the regulation of genes causing MLS\textsubscript{B} resistance. However the variability of the sequence features and the absence of significant sequence similarity unravel the complexity of the task of correctly identifying the leader peptides.
I. INTRODUCTION

A. Ribosome and its role in protein synthesis

Ribosome is a molecular machine designed by nature to make proteins in the cell. During protein synthesis, the genetic code encrypted in the messenger RNA (mRNA) is translated into a polypeptide chain by the ribosome. Ribosomes comprising two subunits, are extremely large complexes composed of many RNA and protein molecules. The prokaryotic ribosome consists of two subunits, the smaller subunit (30S) that interacts with the mRNA and tRNA anticodon stem and the (50S) larger subunit, which interacts with the aminoacyl acceptor end of transfer RNA (tRNA) and the growing polypeptide.

Genetic information is encoded in a series of codons (nucleotide triplets) in the mRNA, which is used as a template by the ribosome for polypeptide synthesis. The small ribosomal subunit acts as the site of interaction between the mRNA and the tRNA molecules. In prokaryotes, the purine rich region upstream of the 5’ end of the mRNA where the smaller ribosomal subunit binds is called the Shine-Dalgarno sequence [1]. Following recognition of the Shine-Dalgarno sequence, the ribosome proceeds to bind the initiator methionyl-tRNA and base pair its anticodon with the AUG codon (start codon) on the mRNA. The initiator tRNA binds the ribosome at the P-site. At the later (elongation) steps of translation, the P-site binds only to the tRNA of the growing polypeptide chain. The aminoacyl-tRNA complementary to the next codon comes to the A-site, which is capable of binding only aminoacyl-tRNAs.
Peptide bond formation takes place in the peptidyl transferase center of the large ribosomal subunit. Following the peptide bond formation, the ribosome shifts to the next codon, moving the new peptidyl tRNA with the growing polypeptide to the P-site, deacylated tRNA into the E-site and making the A-site available for accepting the new tRNA. When the ribosome encounters one of the stop codons namely UAA, UAG or UGA, translation comes to an end and the polypeptide is released from the ribosome and the ribosomal subunits split from each other. Figure 1 shows the structure and functions of the ribosome.

Figure 1. Ribosome structure and function. “P” is peptidyl binding site, “A” is aminoacyl binding site and “E” is Exit site.
B. Regulation of translation by ribosome stalling

The growing polypeptide chain comes out of the ribosome through the nascent peptide exit tunnel, which originates at the peptidyl transferase center (PTC) and runs across the entire length of the large ribosomal subunit (Figure 2). It has been identified that the ribosome has the capability to monitor the nascent peptide structure in the tunnel [2-4]. The tunnel elements interact with the growing polypeptide to cause what is known as ribosome stalling, which in turn is involved in the regulation of gene expression. Several factors trigger the formation of the stalled ribosome complex, including specific physiological conditions (e.g. impaired secretion) or binding of small molecules like an amino acid or an antibiotic. Such ribosome stalling mechanisms contribute to the regulation of expression of several housekeeping genes including those involved in secretion, amino acid metabolism, antibiotic resistance and other functions. The stalling event can be categorized as drug-dependent or drug-independent based on the requirement for the presence of an antibiotic bound to the ribosome.
The bacterial sec and tna operons are the best-characterized regulatory systems operating through drug-independent nascent peptide-directed ribosome stalling [2]. The sec operon is regulated by translation of the leader ORF secM. When secretion activity is sluggish, the ribosome stalls at the end of the secM gene when a specific sequence of the nascent SecM nascent peptide is placed in the ribosome exit tunnel. Switch in mRNA conformation leads to activation of expression of the gene secA, which encodes an important secretion factor. Similarly, the tna operon is regulated by the nascent peptide-controlled stalling of ribosome at the last codon of the leader tnaC cistron. Stalling occurs only at a high concentration of tryptophan and results in transcriptional activation of the tna operon that encodes proteins involved in tryptophan utilization.

Drug-dependent ribosome stalling will be discussed in detail in the following section.
C. Antibiotics and antibiotic resistance

Many antibiotics inhibit protein synthesis. The ribosome is a major target for both natural as well as synthetic antibiotics [5]. Different antibiotics target different sites on the ribosome and inhibit different stages of protein synthesis. Detailed knowledge of the antibiotic binding sites is essential to understand the mechanisms of drug action. Conversely, drugs are excellent tools for studying the functions of the ribosome. Among ribosome targeting antibiotics, macrolides represent an important group. These drugs are among the oldest known protein synthesis inhibitors, which are widely used in clinic and in research. Macrolides are a class of antibiotics that bind the ribosome in the nascent peptide tunnel near the peptidyl transferase center.

Most pathogens become extremely adaptable to antibiotic pressure and hence develop antibiotic resistance. Resistance to MLS₉ antibiotics is conferred by the expression of specific antibiotic resistance genes, which are often induced by the binding of the macrolide antibiotic to the ribosome exit tunnel.

D. Programmed drug-dependent ribosome stalling

The presence of the drug inhibits the growth of the nascent polypeptide when the growing chain reaches the antibiotic binding site, thereby causing translation arrest. With most of the peptides, this leads to peptidyl-tRNA drop off and dissociation of the ribosome from mRNA. But with some specific nascent peptide sequences, translation is arrested with the ribosome retaining peptidyl-tRNA and remaining stably bound to mRNA. This phenomenon is called programmed drug-dependent ribosome stalling,
where the ribosome stalls at a specific codon after the formation of a certain peptide sequence, in the presence of the inducing antibiotic [2, 3, 5]. Short ORFs encoding so-called leader peptides are usually present in the upstream region of the antibiotic resistance gene. The leader ORF plays an important role in regulation of expression of the resistance genes.

E. **Regulation of macrolide resistance genes by ribosome stalling**

One of the best examples of programmed macrolide-dependent ribosome stalling has been identified in MLS$_B$ resistance genes. The most common resistance mechanism operates through modification of the residue A2058 in 23S rRNA by methyltransferase enzymes called Erms. rRNA modification prevents binding of MLS$_B$ antibiotics, thereby causing antibiotic resistance [2, 5, 7]. Macrolide antibiotics induce the gene *ermC*, which is commonly found in *Staphylococcus aureus* and other Gram-positive bacteria. The *ermC* cassette consists of a 19-codon leader ORF *ermCL* with the sequence MGIFSIFVISTVHYQPNKK, which is located 60 bp upstream of the resistance gene *ermC*. Leader peptides are usually named after their downstream genes, along with the suffix ‘L’ to indicate ‘leader’.

![Figure 3. ermC cassette with the leader ORF ermCL](image_url)
The expression of the resistance gene \(ermC\) is controlled by translation attenuation [2, 5]. In the absence of inducing antibiotic, the translation of the resistance gene \(ermC\) is attenuated because the ribosome-binding site (RBS) of \(ermC\) is sequestered in the mRNA stem-loop structure while the leader ORF \(ermCL\) is constitutively translated (Figure 4A). In the presence of inducing antibiotic, the ribosome stalls at the ninth codon of the leader ORF \(ermCL\). Stalling leads to change in mRNA conformation and in turn causes activation of \(ermC\) translation (Figure 4B). Methylation of the 23S rRNA residue A2058 by \(ermC\) causes resistance to the macrolide antibiotics.

The sequence of the ErmCL nascent peptide in the stalled complex is fMGIFSIFVI\(_9\). The C-terminal IFVI sequence has been found to be very important for the formation of the stalled ribosome complex [2, 6]. Mutations in the IFVI segment abolish the formation of stalled complex and result in reduced expression of the \(ermC\) gene [3, 6].
Figure 4. Regulation of expression of \textit{ermC} by translation attenuation.

(A) Uninduced conformation in the absence of antibiotic. (B) Induced conformation in the presence of antibiotic. The mRNA segments involved in the conformation switch are numbered (1-4). The antibiotic bound to the exit tunnel is shown as grey hexagon in (B). (Adapted from [2])
Similar to the regulation of \textit{ermC} by nascent peptide-dependent ribosome stalling, the \textit{ermA} gene is also regulated by leader ORFs. Interestingly, the \textit{ermA} gene has two regulatory leader ORFs, \textit{ermAL1} and \textit{ermAL2} with sequences MCTSIAVVEITLSHS and MGTFSIKFVINKVRYQPNQN respectively (Table 1). In the presence of erythromycin, the ribosome stalling at the first ORF \textit{ermAL1} activates translation of \textit{ermAL2}. Stalling at \textit{ermAL2} activates translation of \textit{ermA}. The sequence of the leader ORF \textit{ermAL2} is similar to that of \textit{ermCL}, with the sequence motif IFVI, suggesting that the ribosome stalls at the second ORF \textit{ermAL2}, thereby causing the resistance gene \textit{ermA} to be expressed [2, 3, 5]. The role of \textit{ermAL1} is not very well established, however experimental evidence demonstrate erythromycin-dependent ribosome stalling at the 8\textsuperscript{th} codon of the \textit{ermAL1} ORF [4, 8] which may increase mRNA stability. Thus, the stabilization of the \textit{ermA} mRNA by ribosome stalling at the leader ORF \textit{ermAL1} and the formation of stalled ribosome complex at the second leader ORF \textit{ermAL2} contribute to the regulation of \textit{ermA} expression [5].

Ribosomal stalling at the tenth codon of the \textit{ermBL} leader ORF encoding peptide MLVFQMRNV\textbf{D}KTSTILKQTKN\textbf{S}DYVDKYVRLIPTSD regulates expression of the gene \textit{ermB}. The critical sequence MRNVD has been identified at the C-terminus of the nascent peptide of the \textit{ermBL} leader ORF [2, 5].

Another class of methyltransferase gene called \textit{ermD} shown to be inducible by macrolide antibiotics contains a leader ORF \textit{ermDL} encoding peptide with sequence MTHSM\textbf{R}LRF\textbf{F}PTLNQ. A second short ORF with an unusual initiator codon UUG has
been identified in the region upstream of the _ermD_ gene. This second ORF named _ermDL2_ encodes the peptide MCMQSKRDQSVLF [5]. However the role of this ORF in _ermD_ regulation has not been studied so far. Table I shows the sequences of other known leader ORFs found in the upstream region of MLS\textsubscript{B} genes.

The mechanisms of regulation of most other macrolide resistance inducible genes are unknown. In most cases, the upstream region of antibiotic resistance genes has not been studied extensively. Therefore, it is essential to analyze the upstream region of the resistance genes to identify the new short peptides, which could be potential regulators of antibiotic resistance. Furthermore, understanding the properties of the stalled ribosome complex critically depend on our knowledge of the structure of the leader peptides. Therefore, knowing if a variety of macrolide resistance genes are controlled by nascent peptide-dependent ribosome stalling and analyzing structures of the corresponding leader peptides could significantly advance our knowledge of molecular mechanisms of translation and regulation of expression of antibiotic resistance genes.

The aim of this project was to identify and compare putative leader peptides of inducible resistance genes. A web browser-based tool with a simple, easy to use interface has been designed to find all possible short ORFs in the upstream region of resistance genes.
TABLE I

SEQUENCES OF KNOWN LEADER PEPTIDES OCCURRING IN THE UPSTREAM REGION OF MACROLIDE RESISTANCE GENES

<table>
<thead>
<tr>
<th>Leader peptide</th>
<th>Sequence</th>
<th>GenBank Accession Number&lt;sup&gt;a&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErmAL1</td>
<td>MCTSIAVVEITLSHS</td>
<td>X03216</td>
</tr>
<tr>
<td>ErmAL2</td>
<td>MGTFISIFVINKVRYQPNQN</td>
<td>X03216</td>
</tr>
<tr>
<td>ErmBL</td>
<td>MLVFQMRNVDKTSTILKQTKNSDYVDKYVRLIPTSD</td>
<td>K00551</td>
</tr>
<tr>
<td>ErmCL</td>
<td>MGIFSIFVISTVHYQPNKK</td>
<td>V01278</td>
</tr>
<tr>
<td>ErmDL</td>
<td>MTHSMRLRFPTLNQ</td>
<td>M29832</td>
</tr>
<tr>
<td>ErmGL1</td>
<td>MNKYSKRDAIN</td>
<td>M15332</td>
</tr>
<tr>
<td>ErmGL2</td>
<td>MGLYSIFVIETVHYQPNEK</td>
<td>M15332</td>
</tr>
<tr>
<td>ErmSL</td>
<td>MSMGIAARPPRAALLPPSVPRSR</td>
<td>M19269</td>
</tr>
<tr>
<td>ErmVL</td>
<td>MAANNAITNSGLGRGCAHSVRMRRGPGALTGPGSHTAR</td>
<td>U59450</td>
</tr>
<tr>
<td>ErmXL</td>
<td>MLISGTAFLRLRNRKAFPTP</td>
<td>M36726</td>
</tr>
<tr>
<td>Erm38L</td>
<td>MSITSMAAPVAAAFIRPRTA</td>
<td>AY154657</td>
</tr>
<tr>
<td>MsrSAL</td>
<td>MTASMRLK</td>
<td>AB016613</td>
</tr>
<tr>
<td>MsrD</td>
<td>MYLIFM</td>
<td>AF274302</td>
</tr>
<tr>
<td>EreAL</td>
<td>MLRSRAVAULKQSYAL</td>
<td>AF0099140</td>
</tr>
</tbody>
</table>

Table Footnotes:

<sup>a</sup> References for each leader peptide and its sequence can be found in the corresponding GenBank entry.
II. MOTIVATION

A. Protein Translation

The study of the upstream region of the antibiotic resistance gene involves deciphering the primary structure of the leader peptides encoded in the putative leader ORFs. The codon table, is used to find the string of amino acids from the nucleotide sequence of the gene.

RNA can be potentially translated in three reading frames. Each of the three reading frames with a different set of codons may be translated to form peptide sequences with different amino acids. Moreover, transcription of the double stranded DNA can happen in either forward or reverse directions, contributing to a total of 6 reading frames. Since transcription and translation of the complementary strand may hypothetically affect protein expression, therefore, for the general use of the tool we were designing, it was good to be able to analyze both strands (6 frames).

B. Sequence features involved in translation

Initiation of translation is the key step in protein synthesis and is a very important step in determining the efficiency of translation. Bacterial translation involves the 30S ribosomal subunit along with the initiation factors (IF) and the fMet-tRNA^Met, which recognize the ribosome binding site (RBS) in the mRNA [9, 10].
The 30S ribosomal subunit and the initiator tRNA bind to the mRNA at the RBS. Usually, the RBS consists of ca. 20 nucleotides upstream from the start codon [11]. It comprises the following sequence elements:

1. Initiation codon – The most common initiation codon is AUG. The codons GUG and UUG are also used as alternative start codons [10].

2. Shine-Dalgarno (SD) sequence – This purine rich sequence 5'-UAAGGAGGU-3' is a part of the RBS. This sequence is located upstream of the initiator codon [9-12].

3. Spacer sequence between the Shine-Dalgarno sequence and the start codon. The distance between these sequence elements influence the efficiency of translation [9]. Figure 5 shows these sequence features seen in the RBS of prokaryotic genes.

4. Sequences immediately downstream from the start codon were proposed to influence translation initiation. However, their role and structural features is poorly defined and were not used in our analysis.

Figure 5. Sequence features in the RBS of a typical bacterial gene. The RBS situated in the upstream region of the gene consists of the Shine-Dalgarno sequence followed by a spacer region and the translation start codon.
C. **Variability of regulatory leader peptides**

Even sketchy analysis of the upstream regions of known antibiotic resistance genes regulated by leader ORFs gives an insight into the extreme variability in these segments. Though the number of known leader peptides is very limited, it is sufficient to define the problem in hand.

Listed below are the features that affect the variability in the leader regions:

1. The leader peptides do not have a specific length. The peptides can be as short as 6 amino acids (*msrD*, GenBank ID AF274302) or as long as 38 amino acids (*ermV*, GenBank ID U59450).

2. Sequence level similarity between known leader peptides is very poor. No specific sequence signatures in the leader peptides have been identified until this study.

3. The distance between the regulatory ORF and the regulated cistron can vary significantly.

4. The sequences of the known regulatory ORFs usually have a defined Shine-Dalgarno sequence in the RBS, which strongly suggests that the leader ORF is translated and leads to the regulation of the downstream gene. However, there are many bacterial genes that are known to be translated efficiently even in the presence of a weak SD or absence of SD sequence so that it is unclear if short ORFs lacking SD region that are present in the leader regions of the resistance genes are translated or not.
5. The distance between the SD and the initiator codon, called the spacer region is highly variable.

6. The leader regions are known to be initiated by alternative start codons like GUG or UUG.

There are many programs and software to find ORFs from the nucleotide sequences. Most of them are however restricted to predicting long ORFs. The known ORF finding programs are unlikely to predict these short leader peptides with reasonable accuracy. Also, not every program is specialized to look for the alternative start codons.
III. METHODS AND IMPLEMENTATION

A. Short ORF finding program

A short ORF (shORF) finding program has been developed with customizable features to identify short ORFs encoding putative leader peptides in the upstream regions of the antibiotic resistance genes.

1. Form Input

The script takes a raw nucleotide sequence or FASTA formatted nucleotide sequence or the GenBank accession number as input. If the GenBank ID is given as input, the program automatically downloads the FASTA sequence of the given accession number from the GenBank nucleotide sequence database (http://www.ncbi.nlm.nih.gov/genbank/) [13]. The start position of the gene can be provided as an optional input.

2. Search Strategy

• Start codons: All possible start codons including AUG, GUG and UUG are searched.

• Search direction: The program looks for shORFs in all 3 reading frames in both forward as well as reverse directions. If the orientation of the resistance gene is known, then only three frames, co-linear with the resistance ORF need to be considered. However, in order to expand the applicability of the shORF finder, we included the six-frame search option.
• **Search space:** If gene start position is given, the program searches for shORFs 500 bases (or the available bases) upstream of the given start site. Else, the program searches for all available shORFs in the given sequence.

• **Stop codons:** The short peptides terminating in any one of the 3 stop codons namely UAG, UGA and UAA are identified and reported.

• **SD-sequence:** The script is designed to look for the presence of SD sequence with the consensus UAAGGAGGU. The search algorithm looks for exact match to all sub-sequences of the SD consensus, starting from length 4. The longest sub-sequence identified in the query sequence is reported. Examples of smaller SD sequence include AGGAG, GGAG, etc. Additionally, the script also reports those short peptides without a defined SD sequence, which is indicated clearly in the result.

• **Spacer sequence:** The algorithm allows 3-20 nucleotides between the putative SD sequence and the start codon.

3. **Search Results**

The search results are displayed in a tabular format, with buttons to show the forward or reverse strands. The table gives all information about the short peptide sequence including the nucleotide and peptide sequence of the shORF, SD sequence, length of SD sequence, position of SD, spacer sequence and length, start codon, sequence upstream of start codon, length and position of the shORF with respect to the given query sequence, length of the peptide and the distance between the shORF and the gene if the gene start is given.
There are separate tables in the results section, which include ORFs without a well-defined SD sequence. Though this software was developed for finding all short peptides in the upstream regions of antibiotic resistance genes, it would work well with any nucleotide sequence.

**B. Software**

The web-based user interface was developed using HTML, JavaScript and DHTML programming languages. The website was setup using Apache 2.0 web server. Scripts and modules to identify the short ORFs were written in Perl-CGI. Figures 6 shows the front page of the shORF finding program. A screen shot of the search results is shown in Figure 7. The tool is available at: [http://mankinlab.cpb.uic.edu/orf.html](http://mankinlab.cpb.uic.edu/orf.html)

The HTML code (*orf.html*), CGI script (*shine.cgi*) and Perl modules (*PerlModules.pm*, *orf_finder.pm* and *w_orf_finder.pm*) are given in Appendix.
Figure 6. Screen-shot of shORF finder search form. The GenBank accession number or the nucleotide raw or FASTA sequence can be entered in the boxes provided. The start position of the resistance gene can be entered optionally.
Figure 7. Search results showing all possible short peptides of length 4-50 amino acids that can be translated from the query nucleotide sequence. The shORFs in both forward and reverse orientations are shown. The search also generates the list of shORFs without a SD sequence in the upstream region.
IV. RESULTS AND DISCUSSION

The main aim of the project was to computationally identify and classify the short open reading frames which act as leader peptides in the upstream regions of antibiotic resistance genes, thereby regulating expression.

A. Dataset used in the analysis

A website maintained by Dr. Marilyn Roberts at University of Washington (http://faculty.washington.edu/marilynr/) contains known MLS antibiotic resistance genes. This analysis was started with a list of all known rRNA methylase genes, efflux and inactivating genes from Roberts’ website [14]. The leader peptides of some of these genes have been reported in literature while most others have not been analyzed so far.

A comprehensive list of 77 rRNA methylase genes (sequences of 33 classes of \textit{erm} genes along with the variants from different plasmids / transposons deposited in GenBank) was collected from Roberts’ website. The upstream region of each sequence was analyzed using the shORF finder program to find all possible short ORFs of length up to 150 nucleotides. The potential leader ORFs identified by the program was manually scanned and the putative leader ORFs were tabulated. Of the 77 genes studied, 60 upstream sequences were identified to have putative leader ORFs, including those reported previously and those hypothetical ORFs predicted by the program. The program was able to identify all 36 leader ORFs that have been described earlier, along with 24 new putative leader ORFs.
B. Motif groups in leader peptides

Sequence analysis of the known and putative leader peptides revealed that the shORFs could be classified based on the presence of common sequence motifs. Identification of consensus motifs was driven in part by the experimental data obtained in the lab in parallel with this study. Since the consensus motifs appear to be very short and the total number of sequences was very small, none of the available motif-searching algorithms was able to identify them. Therefore, consensus was found by manual scanning of the sequences. The length of the ORFs range between 6-50 amino acids and the motifs found in these sequences were around 3-4 amino acids long.

The leader peptides identified so far can be grouped into 4 major classes: IFVI group, RLR group, IAVV group and the miscellaneous group. The motif classification of all known and putative leader peptides is shown in Table II.

Table II shows the list of all the key features of known and newly identified putative leader peptides extracted from the results of the shORF program like the start codon, presence or absence of SD sequence, actual SD sequence, spacer distance between the ribosome binding site and the start of the putative leader ORF, number of nucleotides between the leader peptide and the gene. The list of all genes, the known and predicted leader ORF sequences and the GenBank accession numbers of the gene sequences is given in Table II. The list of potential short ORFs obtained from the shORF program was manually scanned to select the candidate leaders.
# TABLE II

LEADER PEPTIDES OF MLS<sub>B</sub> RESISTANCE GENES REPORTED IN LITERATURE AND IDENTIFIED BY THE shORF FINDING PROGRAM

<table>
<thead>
<tr>
<th>Name&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Leader peptide sequence&lt;sup&gt;b&lt;/sup&gt;</th>
<th>GenBank accession number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IAVV peptides</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ErmAL&lt;sup&gt;c&lt;/sup&gt;1</td>
<td>MCTIAVVDITLSHL</td>
<td>AF002716</td>
</tr>
<tr>
<td>ErmAL&lt;sup&gt;c&lt;/sup&gt;1</td>
<td>MCTIAVVEITLSHS</td>
<td>X03216</td>
</tr>
<tr>
<td>Erm36L</td>
<td>MGPSIAVTRFRRF</td>
<td>AF462611</td>
</tr>
<tr>
<td><strong>IFVI peptides</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ErmAL&lt;sup&gt;c&lt;/sup&gt;2</td>
<td>MGMFSIFVIERFHYQPNQK</td>
<td>AF002716</td>
</tr>
<tr>
<td>ErmAL&lt;sup&gt;c&lt;/sup&gt;2</td>
<td>MGTFIFVINKRYQPNQN</td>
<td>X03216</td>
</tr>
<tr>
<td>ErmCL&lt;sup&gt;c&lt;/sup&gt;</td>
<td>MGIFSIFVISTHYQPNKK</td>
<td>V01278</td>
</tr>
<tr>
<td>ErmGL2</td>
<td>MGLYSIFVETVHYQPNEK</td>
<td>M15332</td>
</tr>
<tr>
<td>ErmTL</td>
<td>MGIFSIFVINTVHYQPNKK</td>
<td>M64090</td>
</tr>
<tr>
<td>ErmYL</td>
<td>MGNCSLFVINTVHYQPNEK</td>
<td>AB014481</td>
</tr>
<tr>
<td>Erm33L</td>
<td>MGIFSIFVINTVHYQPNKK</td>
<td>AJ313523</td>
</tr>
<tr>
<td><strong>RLR peptides</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EreAL</td>
<td>ML&lt;sup&gt;R&lt;/sup&gt;SRAVALKQSYAL</td>
<td>AF099140</td>
</tr>
<tr>
<td>Erm34L</td>
<td>MH&lt;sup&gt;R&lt;/sup&gt;FIRLFVLNK</td>
<td>AY234334</td>
</tr>
<tr>
<td>ErmDL</td>
<td>MTHSMRLRFPTLNQ</td>
<td>M29832</td>
</tr>
<tr>
<td>MefAL</td>
<td>MTASMRLR</td>
<td>AF274302</td>
</tr>
<tr>
<td>MsrAL</td>
<td>MTASMRLK</td>
<td>AB016613</td>
</tr>
<tr>
<td>MsrCL</td>
<td>MTASMKLFELLNNN</td>
<td>AY004350</td>
</tr>
<tr>
<td>Erm39L</td>
<td>MSVTTYIRLRIT</td>
<td>AY487229</td>
</tr>
<tr>
<td>ErmXL</td>
<td>MLISGTAFLRLRTNRKAFPTP</td>
<td>M36726</td>
</tr>
<tr>
<td>Name(^a)</td>
<td>Leader peptide sequence(^b)</td>
<td>GenBank accession number</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>ErmQL</td>
<td>MIMNGGIA_IRLRR</td>
<td>L22689</td>
</tr>
<tr>
<td>EreAL</td>
<td>MTPNSFK_TPLRGAA</td>
<td>AY183453</td>
</tr>
<tr>
<td>ErmFL</td>
<td>MKPTGLGSISQRVTLVK</td>
<td>M17808</td>
</tr>
<tr>
<td>ErmWL</td>
<td>MGFSFTGSAFIRLRTA</td>
<td>D14532</td>
</tr>
<tr>
<td><strong>Miscellaneous peptides</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MetBL</td>
<td>MYLIFM</td>
<td>FJ196385</td>
</tr>
<tr>
<td>MsrDL</td>
<td>MYLIFM</td>
<td>AF274302</td>
</tr>
<tr>
<td>ErmGL1</td>
<td>MRIDDYCS</td>
<td>L42817</td>
</tr>
<tr>
<td>MphCL</td>
<td>MYQIKNGN</td>
<td>AF167161</td>
</tr>
<tr>
<td>EreAL</td>
<td>MSLVIGEAKV</td>
<td>AF512546</td>
</tr>
<tr>
<td>Erm37L</td>
<td>MRTAPEPWGW</td>
<td>BX842578</td>
</tr>
<tr>
<td>MphBL</td>
<td>MAKEALEVQGS</td>
<td>D85892</td>
</tr>
<tr>
<td>ErmGL1</td>
<td>MNKYSKRDAIN</td>
<td>M15332</td>
</tr>
<tr>
<td>ErmEL</td>
<td>MRVSVRVAACARC</td>
<td>M11200</td>
</tr>
<tr>
<td>ErmFL</td>
<td>MMLCCRLSSFLLSR</td>
<td>M62487</td>
</tr>
<tr>
<td>MphAL</td>
<td>MNKTKGCLIANFATVPD</td>
<td>D16251</td>
</tr>
<tr>
<td>Erm38L</td>
<td>MSITSMAAPVAAFIRP_RTA</td>
<td>AY154657</td>
</tr>
<tr>
<td>EreAL</td>
<td>MQLTVKSFVRFACYASYRN</td>
<td>AF512546</td>
</tr>
<tr>
<td>ErmGL2</td>
<td>MNHEYVLFSKNINIRKEMQ</td>
<td>L42817</td>
</tr>
<tr>
<td>EreBL</td>
<td>MRIXRTAYARPCALAEGERX</td>
<td>A15097</td>
</tr>
<tr>
<td>EreBL</td>
<td>MRINRKTYARPCALAEEGRG</td>
<td>AB207867</td>
</tr>
<tr>
<td>Erm41L</td>
<td>MMVLRRVRPTVATPVGLVSAH</td>
<td>EU177504</td>
</tr>
<tr>
<td>ErmSL</td>
<td>MSMSGIAAPPRAALLPPPSVPRSR</td>
<td>M19269</td>
</tr>
<tr>
<td>ErmBL</td>
<td>MLVFQMCNVDKSTV~LQTKNSDYADK</td>
<td>U86375</td>
</tr>
<tr>
<td>Name</td>
<td>Leader peptide sequence</td>
<td>GenBank accession number</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>ErmBL</td>
<td>MLVFQMRNVDDKTSTVLKQTKNSDYADK</td>
<td>M11180</td>
</tr>
<tr>
<td>ErmTL</td>
<td>MRNVDKTSTVLKQTKNSDYADK</td>
<td>AJ488494</td>
</tr>
<tr>
<td>ErmBL</td>
<td>MLVFQMRNVDDKTSTVLKQTKNSDLRR</td>
<td>AF299292</td>
</tr>
<tr>
<td>ErmBL</td>
<td>MLVFQIRNVDDKTSTGLKQTKNSDYADK</td>
<td>AF080450</td>
</tr>
<tr>
<td>ErmBL</td>
<td>MLVFQMRNVDKTSTILKQTKNSDYVDKYVRLIPTSD</td>
<td>K00551</td>
</tr>
<tr>
<td>ErmBL</td>
<td>MLVFQMYQMDKTSTVLKQTKKKSDYADK</td>
<td>M19270</td>
</tr>
<tr>
<td>ErmFL</td>
<td>MLSAFIFSSFLIYRAKLLNLPLYNYKRISL</td>
<td>M62487</td>
</tr>
<tr>
<td>ErmNL</td>
<td>MARTLFAGRTELWAPAIEPPVKAATHTAVRRD</td>
<td>X97721</td>
</tr>
<tr>
<td>ErmVL</td>
<td>MAANNAITNSGLGRGCAHSMRMRPGALTGPHTAR</td>
<td>U59450</td>
</tr>
</tbody>
</table>

**Table footnotes:**

- **a** Leader peptides have been designated the suffix ‘L’. Numbers 1 and 2 refer to the 5’-proximal and the consecutive peptides on the same cistron, respectively.

- **b** Leader peptides have been grouped according to similarity in amino acid sequence. Within each group, the common motif has been underlined.

- **c** For ermC [3] and ermA [4, 8] these motifs have also been shown to be important for ribosome stalling. The codons corresponding to the amino acids in bold have been experimentally shown to be present in the P site of the stalled ribosome complex (Ramu et al., unpublished results).
C. Resistance profile and mechanism of antibiotic resistance genes

The analysis of upstream regions of the MLS\textsubscript{B} resistance methylase genes was then expanded to include other genes conferring macrolide resistance including genes encoding drug-inactivating enzymes (such as esterases, phosphorlyases and others) and efflux pumps. An extensive literature survey of the inducibility and the antibiotic responsible for induction of the macrolide resistance genes was done. The information on mechanism of regulation of gene expression, if known, was also gathered from the literature. The results of this study are tabulated in Table III.

About half of the genes were reported to be inducible. Most of the others were reported to exhibit constitutive expression, while the inducibility of a few was unknown. In general, macrolide antibiotics act as inducers while lincosamides and streptogramins fall in the category of non-inducing antibiotics. Exposure of cells with an inducible gene to the non-inducing antibiotics (such as lincosamides or streptogramins) leads to selection for alterations that convert inducible resistance gene to constitutive. Such a change in preference from inducible to constitutive expression of the resistance gene is caused by mutations or indels in the regions responsible for inducibility. Thus the genes marked as constitutive in Table III may have had inducible predecessors.

Since more than half of the genes in Table III are either known or suspected to be inducible, it is reasonable to think that the mechanism of regulation is controlled by short regulatory peptides similar to that of the leader ORFs in \textit{ermC}, \textit{ermA} and other methylase genes discussed earlier. The shORF program was used to check for the presence of potential regulatory ORFs in each gene.
<table>
<thead>
<tr>
<th>Gene Name</th>
<th>Resistance mechanism</th>
<th>Organism</th>
<th>Resistance profile</th>
<th>Inducibility</th>
<th>GenBank Accession Number</th>
<th>Reference – Pubmed ID</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ermA</em></td>
<td>Target site modification</td>
<td><em>Staphylococcus aureus</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>X03216</td>
<td>3004956</td>
</tr>
<tr>
<td><em>ermA</em></td>
<td>Target site modification</td>
<td><em>Streptococcus pyogenes</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>AF002716</td>
<td>9527769</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Target site modification</td>
<td><em>Escherichia coli</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Constitutive</td>
<td>M19270</td>
<td>2832378</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Target site modification</td>
<td><em>Streptococcus sanguinis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>K00551</td>
<td>6406429</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Target site modification</td>
<td><em>Enterococcus faecalis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>M11180</td>
<td>2997130</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Target site modification</td>
<td><em>Lactobacillus reuteri</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Unknown</td>
<td>AF080450</td>
<td>10413663</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Target site modification</td>
<td><em>Staphylococcus intermedius</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible&lt;sup&gt;b&lt;/sup&gt;</td>
<td>AF299292</td>
<td>11230937</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Target site modification</td>
<td><em>Enterococcus faecalis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>U86375</td>
<td>9791136</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Resistance mechanism</td>
<td>Organism</td>
<td>Resistance profile&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Inducibility</td>
<td>GenBank Accession Number</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>------------------------</td>
<td>-------------------------------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>ermC</td>
<td>Target site modification</td>
<td><em>Staphylococcus simulans</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Constitutive</td>
<td>AF019140</td>
<td>9742700</td>
</tr>
<tr>
<td>ermC</td>
<td>Target site modification</td>
<td><em>Staphylococcus aureus</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>V01278</td>
<td>6279574</td>
</tr>
<tr>
<td>ermD</td>
<td>Target site modification</td>
<td><em>Bacillus licheniformis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>M77505</td>
<td>1713206</td>
</tr>
<tr>
<td>ermD</td>
<td>Target site modification</td>
<td><em>Bacillus anthracis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>L08389</td>
<td>8473865</td>
</tr>
<tr>
<td>ermD</td>
<td>Target site modification</td>
<td><em>Bacillus licheniformis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>M29832</td>
<td>6429477</td>
</tr>
<tr>
<td>ermE</td>
<td>Target site modification</td>
<td><em>Saccharopolyspora erythraea</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Unknown</td>
<td>M11200</td>
<td>3934045, 2998943</td>
</tr>
<tr>
<td>ermF</td>
<td>Target site modification</td>
<td><em>Bacteroides fragilis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Unknown</td>
<td>M17808</td>
<td>2820936</td>
</tr>
<tr>
<td>ermF</td>
<td>Target site modification</td>
<td><em>Bacteroides fragilis</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Unknown</td>
<td>M62487</td>
<td>1905805</td>
</tr>
<tr>
<td>ermG</td>
<td>Target site modification</td>
<td><em>Lysinibacillus sphaericus</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>M15332</td>
<td>3025178</td>
</tr>
<tr>
<td>ermG</td>
<td>Target site modification</td>
<td><em>Bacteroides thetaiotaomicron</em></td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Unknown&lt;sup&gt;c&lt;/sup&gt;</td>
<td>L42817</td>
<td>8834912</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Resistance mechanism</td>
<td>Organism</td>
<td>Resistance profile</td>
<td>Inducibility</td>
<td>GenBank Accession Number</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>----------</td>
<td>--------------------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>ermH</td>
<td>Target site modification</td>
<td><em>Streptomyces thermotolerans</em></td>
<td>MLSB</td>
<td>Inducible</td>
<td>M16503</td>
<td>3036668</td>
</tr>
<tr>
<td>ermN</td>
<td>Target site modification</td>
<td><em>Streptomyces fradiae</em></td>
<td>MLSB</td>
<td>Constitutive</td>
<td>X97721</td>
<td>8973363, 1995426</td>
</tr>
<tr>
<td>ermO</td>
<td>Target site modification</td>
<td><em>Streptomyces ambofaciens</em></td>
<td>MLSB</td>
<td>Unknown</td>
<td>AJ223970</td>
<td>10517588</td>
</tr>
<tr>
<td>ermQ</td>
<td>Target site modification</td>
<td><em>Clostridium perfringens</em></td>
<td>MLSB</td>
<td>Unknown</td>
<td>L22689</td>
<td>8067735</td>
</tr>
<tr>
<td>ermR</td>
<td>Target site modification</td>
<td><em>Arthrobacter sp.</em></td>
<td>M</td>
<td>Unknown</td>
<td>M11276</td>
<td>4043733</td>
</tr>
<tr>
<td>ermS</td>
<td>Target site modification</td>
<td><em>Streptomyces fradiae</em></td>
<td>MLSB</td>
<td>Inducible</td>
<td>M19269</td>
<td>3127381</td>
</tr>
<tr>
<td>ermT</td>
<td>Target site modification</td>
<td><em>Lactobacillus reuteri</em></td>
<td>MLSB</td>
<td>Constitutive</td>
<td>M64090</td>
<td>8171126</td>
</tr>
<tr>
<td>ermT</td>
<td>Target site modification</td>
<td><em>Lactobacillus fermentum</em></td>
<td>MLSB</td>
<td>Constitutive</td>
<td>AJ488494</td>
<td>14597008</td>
</tr>
<tr>
<td>ermV</td>
<td>Target site modification</td>
<td><em>Streptomyces viridochromogenes</em></td>
<td>MLSB</td>
<td>Inducible</td>
<td>U59450</td>
<td>6163765, 9055987</td>
</tr>
<tr>
<td>ermW</td>
<td>Target site modification</td>
<td><em>Micromonospora griseorubida</em></td>
<td>MLSB</td>
<td>Unknown</td>
<td>D14532</td>
<td>8163173</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Resistance mechanism</td>
<td>Organism</td>
<td>Resistance profile&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Inducibility</td>
<td>GenBank Accession Number</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>----------</td>
<td>-------------------------------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>ermX</td>
<td>Target site modification</td>
<td>Corynebacterium diphtheriae</td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>M36726</td>
<td>11408212</td>
</tr>
<tr>
<td>ermY</td>
<td>Target site modification</td>
<td>Staphylococcus aureus</td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>AB014481</td>
<td>11751136</td>
</tr>
<tr>
<td>erm30</td>
<td>Target site modification</td>
<td>Streptomyces venezuelae</td>
<td>Unknown</td>
<td>Unknown</td>
<td>AF079138</td>
<td>9770448</td>
</tr>
<tr>
<td>erm31</td>
<td>Target site modification</td>
<td>Streptomyces venezuelae</td>
<td>Unknown</td>
<td>Unknown</td>
<td>AF079138</td>
<td>9770448</td>
</tr>
<tr>
<td>erm32</td>
<td>Unknown&lt;sup&gt;d&lt;/sup&gt;</td>
<td>Streptomyces fradiae</td>
<td>M</td>
<td>Inducible</td>
<td>AJ009971</td>
<td>10348045</td>
</tr>
<tr>
<td>erm33</td>
<td>Target site modification</td>
<td>Staphylococcus sciuri</td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>AJ313523</td>
<td>12384375</td>
</tr>
<tr>
<td>erm34</td>
<td>Target site modification</td>
<td>Bacillus clausii</td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Unknown</td>
<td>AY234334</td>
<td>14711653</td>
</tr>
<tr>
<td>erm35</td>
<td>Target site modification</td>
<td>Bacteroides coprospuis</td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Unknown</td>
<td>AF319779</td>
<td>-</td>
</tr>
<tr>
<td>erm36</td>
<td>Target site modification</td>
<td>Micrococcus luteus</td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>AF462611</td>
<td>12177341</td>
</tr>
<tr>
<td>erm37</td>
<td>Target site modification</td>
<td>Mycobacterium tuberculosis</td>
<td>MLS&lt;sub&gt;B&lt;/sub&gt;</td>
<td>Inducible</td>
<td>Z74025, BX842578</td>
<td>9634230, 16174779</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Resistance mechanism</td>
<td>Organism</td>
<td>Resistance profile(^a)</td>
<td>Inducibility</td>
<td>GenBank Accession Number</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>----------</td>
<td>--------------------------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>erm38</td>
<td>Target site modification</td>
<td><em>Mycobacterium smegmatis</em></td>
<td>ML</td>
<td>Inducible</td>
<td>AY154657, CP000480</td>
<td>14506008, 16127056</td>
</tr>
<tr>
<td>erm39</td>
<td>Target site modification</td>
<td><em>Mycobacterium fortuitum</em></td>
<td>ML</td>
<td>Inducible</td>
<td>AY487229</td>
<td>15590712</td>
</tr>
<tr>
<td>erm40</td>
<td>Target site modification</td>
<td><em>Mycobacterium mageritense</em></td>
<td>ML</td>
<td>Constitutive</td>
<td>AY570506</td>
<td>17005837</td>
</tr>
<tr>
<td>erm41</td>
<td>Target site modification</td>
<td><em>Mycobacterium abscessus</em></td>
<td>ML</td>
<td>Inducible</td>
<td>EU177504</td>
<td>19171799</td>
</tr>
<tr>
<td>msrA</td>
<td>Efflux</td>
<td><em>Staphylococcus epidermidis</em></td>
<td>MS(_B)</td>
<td>Inducible</td>
<td>X52085</td>
<td>2233255</td>
</tr>
<tr>
<td>msrC</td>
<td>Efflux</td>
<td><em>Enterococcus faecium</em></td>
<td>MS(_B)</td>
<td>Inducible</td>
<td>AY004350</td>
<td>11120975</td>
</tr>
<tr>
<td>msrD</td>
<td>Efflux</td>
<td><em>Streptococcus pneumoniae</em></td>
<td>M</td>
<td>Inducible</td>
<td>AF274302</td>
<td>11398110, 16223938</td>
</tr>
<tr>
<td>oleB</td>
<td>Efflux</td>
<td><em>Streptomyces antibioticus</em></td>
<td>Oleandomycin</td>
<td>Unknown</td>
<td>L36601</td>
<td>7565095</td>
</tr>
<tr>
<td>oleC</td>
<td>Efflux</td>
<td><em>Streptomyces antibioticus</em></td>
<td>Oleandomycin</td>
<td>Unknown</td>
<td>L06249</td>
<td>8326867</td>
</tr>
<tr>
<td>srmB</td>
<td>Efflux</td>
<td><em>Streptomyces ambofaciens</em></td>
<td>Spiramycin</td>
<td>Inducible</td>
<td>X63451</td>
<td>1508047</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Resistance mechanism</td>
<td>Organism</td>
<td>Resistance profile</td>
<td>Inducibility</td>
<td>GenBank Accession Number</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>----------</td>
<td>-------------------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>tlrC</td>
<td>Efflux</td>
<td><em>Streptomyces fradiae</em></td>
<td>Tylosin</td>
<td>Inducible</td>
<td>M57437</td>
<td>1864505</td>
</tr>
<tr>
<td>mefA</td>
<td>Efflux</td>
<td><em>Streptococcus pneumoniae</em></td>
<td>ML</td>
<td>Unknown</td>
<td>AF274302</td>
<td>11398110</td>
</tr>
<tr>
<td>mefB</td>
<td>Efflux</td>
<td><em>Escherichia coli</em></td>
<td>ML</td>
<td>Unknown</td>
<td>FJ196385</td>
<td>19131424</td>
</tr>
<tr>
<td>ereA</td>
<td>Drug modification</td>
<td><em>Providencia stuartii</em></td>
<td>M</td>
<td>Unknown</td>
<td>AF099140</td>
<td>12654734</td>
</tr>
<tr>
<td>ereA</td>
<td>Drug modification</td>
<td><em>Escherichia coli</em></td>
<td>MLSB</td>
<td>Constitutive</td>
<td>AY183453</td>
<td>14506050</td>
</tr>
<tr>
<td>ereA</td>
<td>Drug modification</td>
<td><em>Vibrio cholerae</em></td>
<td>M</td>
<td>Unknown</td>
<td>AF512546</td>
<td>12183252</td>
</tr>
<tr>
<td>ereB</td>
<td>Drug modification</td>
<td><em>Escherichia coli</em></td>
<td>M</td>
<td>Constitutive</td>
<td>X03988</td>
<td>3523438</td>
</tr>
<tr>
<td>ereB</td>
<td>Drug modification</td>
<td><em>Escherichia coli</em></td>
<td>M</td>
<td>Constitutive</td>
<td>AB207867.1</td>
<td>2546492</td>
</tr>
<tr>
<td>ereB</td>
<td>Drug modification</td>
<td><em>Escherichia coli</em></td>
<td>M</td>
<td>Unknown</td>
<td>A15097</td>
<td>-</td>
</tr>
<tr>
<td>mphA</td>
<td>Drug modification</td>
<td><em>Escherichia coli</em></td>
<td>M</td>
<td>Inducible</td>
<td>D16251</td>
<td>8619599, 10960087</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Resistance mechanism</td>
<td>Organism</td>
<td>Resistance profilea</td>
<td>Inducibility</td>
<td>GenBank Accession Number</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>-----------</td>
<td>---------------------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>mphB</td>
<td>Drug modification</td>
<td><em>Escherichia coli</em></td>
<td>M</td>
<td>Constitutive</td>
<td>D85892</td>
<td>8900063</td>
</tr>
<tr>
<td>mphC</td>
<td>Drug modification</td>
<td><em>Staphylococcus aureus</em></td>
<td>M</td>
<td>Unknown</td>
<td>AF167161</td>
<td>-</td>
</tr>
<tr>
<td>mphD</td>
<td>Drug modification</td>
<td><em>Pseudomonas aeruginosa</em></td>
<td>M</td>
<td>Unknown</td>
<td>AB048591</td>
<td>11184231</td>
</tr>
</tbody>
</table>

**Table footnotes:**

a Abbreviations key: M - Macrolides; ML - Macrolides and Lincosamides; MSB - Macrolides and Streptogramins B; MLSB - Macrolides, Lincosamides and Streptogramins B.

b Both inducible and constitutive isolates of this strain were found.

c Reported as “Likely to be constitutive”.

d Though the encoded protein is similar to rRNA methylases, it does not methylate rRNA. The mechanism of resistance could involve drug modification.
### TABLE IV
SEQUENCE FEATURES OF KNOWN AND PREDICTED LEADER ORFs OF ANTIBIOTIC RESISTANCE GENES

<table>
<thead>
<tr>
<th>Gene Name</th>
<th>Inducibility</th>
<th>Annotated leader peptide</th>
<th>Putative Shine-Dalgarno sequence</th>
<th>Leader ORF start codon</th>
<th>Leader ORF length (amino acid residues)</th>
<th>Distance between leader ORF &amp; resistance gene (bp)</th>
<th>Reference – Pubmed ID</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ermA</em></td>
<td>Inducible</td>
<td>Yes (LP2)</td>
<td>AAGGAG</td>
<td>AUG</td>
<td>19</td>
<td>57</td>
<td>3004956</td>
</tr>
<tr>
<td><em>ermA</em></td>
<td>Inducible</td>
<td>Yes (LP2)</td>
<td>AAGGAGGU</td>
<td>AUG</td>
<td>19</td>
<td>57</td>
<td>9527769</td>
</tr>
<tr>
<td><em>ermA</em></td>
<td>Inducible</td>
<td>Yes (LP1)</td>
<td>UAAGGAGG</td>
<td>AUG</td>
<td>15</td>
<td>143</td>
<td>9527769</td>
</tr>
<tr>
<td><em>ermA</em></td>
<td>Inducible</td>
<td>Yes (LP1)</td>
<td>UAAGGAGGU</td>
<td>AUG</td>
<td>15</td>
<td>144</td>
<td>3004956</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Constitutive</td>
<td>Yes</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>32</td>
<td>125</td>
<td>2832378</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Inducible</td>
<td>Yes</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>36</td>
<td>92</td>
<td>6406429</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Inducible</td>
<td>Yes</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>27</td>
<td>124</td>
<td>2997130, 10413663</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Inducible</td>
<td>Yes</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>26</td>
<td>128</td>
<td>11230937</td>
</tr>
<tr>
<td><em>ermB</em></td>
<td>Inducible</td>
<td>Yes</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>27</td>
<td>124</td>
<td>9791136</td>
</tr>
<tr>
<td><em>ermC</em></td>
<td>Constitutive</td>
<td>Yes</td>
<td>UAAGGAGG</td>
<td>AUG</td>
<td>19</td>
<td>121</td>
<td>9742700</td>
</tr>
<tr>
<td><em>ermC</em></td>
<td>Inducible</td>
<td>Yes</td>
<td>UAAGGAGG</td>
<td>AUG</td>
<td>19</td>
<td>60</td>
<td>6279574</td>
</tr>
<tr>
<td><em>ermD</em></td>
<td>Inducible</td>
<td>Yes</td>
<td>AGGAGG</td>
<td>AUG</td>
<td>14</td>
<td>276</td>
<td>1713206, 8473865, 6429477</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Inducibility</td>
<td>Annotated leader peptide</td>
<td>Putative Shine-Dalgarano sequence</td>
<td>Leader ORF start codon</td>
<td>Leader ORF length (amino acid residues)</td>
<td>Distance between leader ORF &amp; resistance gene (bp)</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------------------------</td>
<td>------------------------</td>
<td>------------------------------------------</td>
<td>-------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ermE</td>
<td>Unknown</td>
<td>No</td>
<td>AGGA</td>
<td>AUG</td>
<td>13</td>
<td>146</td>
<td>3934045, 2998943</td>
</tr>
<tr>
<td>ermF</td>
<td>Unknown</td>
<td>No</td>
<td>GGUG</td>
<td>AUG</td>
<td>20</td>
<td>152</td>
<td>2820936</td>
</tr>
<tr>
<td>ermF</td>
<td>Unknown</td>
<td>No</td>
<td>AGGU</td>
<td>AUG</td>
<td>31</td>
<td>35</td>
<td>1905805</td>
</tr>
<tr>
<td>ermF</td>
<td>Unknown</td>
<td>No</td>
<td>-</td>
<td>GUG</td>
<td>14</td>
<td>94</td>
<td>1905805</td>
</tr>
<tr>
<td>ermG</td>
<td>Inducible</td>
<td>Yes (LP2)</td>
<td>UAAGGAGGU</td>
<td>AUG</td>
<td>19</td>
<td>56</td>
<td>3025178</td>
</tr>
<tr>
<td>ermG</td>
<td>Unknown&lt;sup&gt;5&lt;/sup&gt;</td>
<td>LP1</td>
<td>-</td>
<td>AUG</td>
<td>8</td>
<td>122</td>
<td>8834912</td>
</tr>
<tr>
<td>ermG</td>
<td>Inducible</td>
<td>LP1</td>
<td>AGAGG</td>
<td>AUG</td>
<td>11</td>
<td>-</td>
<td>3025178</td>
</tr>
<tr>
<td>ermG</td>
<td>Unknown&lt;sup&gt;5&lt;/sup&gt;</td>
<td>LP2</td>
<td>-</td>
<td>AUG</td>
<td>19</td>
<td>17</td>
<td>8834912</td>
</tr>
<tr>
<td>ermN</td>
<td>Constitutive</td>
<td>No</td>
<td>GAGG</td>
<td>UUG</td>
<td>32</td>
<td>17</td>
<td>8973363, 1995426</td>
</tr>
<tr>
<td>ermQ</td>
<td>Unknown</td>
<td>No</td>
<td>AGGGAGG</td>
<td>AUG</td>
<td>14</td>
<td>80</td>
<td>8067735</td>
</tr>
<tr>
<td>ermS</td>
<td>Inducible</td>
<td>Yes</td>
<td>AGAGGUG</td>
<td>GUG</td>
<td>24</td>
<td>227</td>
<td>3127381</td>
</tr>
<tr>
<td>ermT</td>
<td>Constitutive</td>
<td>Yes</td>
<td>AAGGGAGG</td>
<td>AUG</td>
<td>19</td>
<td>85</td>
<td>8171126</td>
</tr>
<tr>
<td>ermT</td>
<td>Constitutive</td>
<td>Yes (truncated LP)</td>
<td>AAGGGAGG</td>
<td>AUG</td>
<td>22</td>
<td>126</td>
<td>14597008</td>
</tr>
<tr>
<td>ermV</td>
<td>Inducible</td>
<td>Yes</td>
<td>-</td>
<td>AUG</td>
<td>38</td>
<td>102</td>
<td>6163765, 9055987</td>
</tr>
<tr>
<td>ermW</td>
<td>Unknown</td>
<td>No</td>
<td>AGGA</td>
<td>AUG</td>
<td>39</td>
<td>187</td>
<td>8163173</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Inducibility</td>
<td>Annotated leader peptide</td>
<td>Putative Shine-Dalgaranno sequence</td>
<td>Leader ORF start codon</td>
<td>Leader ORF length (amino acid residues)</td>
<td>Distance between leader ORF &amp; resistance gene (bp)</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------------------------</td>
<td>------------------------</td>
<td>----------------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>ermX</td>
<td>Inducible</td>
<td>Yes</td>
<td>AAGU</td>
<td>UUG</td>
<td>21</td>
<td>80</td>
<td>11408212</td>
</tr>
<tr>
<td>ermY</td>
<td>Inducible</td>
<td>Yes</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>19</td>
<td>60</td>
<td>11751136</td>
</tr>
<tr>
<td>erm33</td>
<td>Inducible</td>
<td>No</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>19</td>
<td>60</td>
<td>12384375</td>
</tr>
<tr>
<td>erm34</td>
<td>Unknown</td>
<td>Yes</td>
<td>GGAGG</td>
<td>AUG</td>
<td>13</td>
<td>232</td>
<td>14711653</td>
</tr>
<tr>
<td>erm36</td>
<td>Inducible</td>
<td>Yes</td>
<td>-</td>
<td>AUG</td>
<td>14</td>
<td>98</td>
<td>12177341</td>
</tr>
<tr>
<td>erm37</td>
<td>Inducible</td>
<td>No</td>
<td>GGAG</td>
<td>AUG</td>
<td>10</td>
<td>98</td>
<td>9634230, 16174779</td>
</tr>
<tr>
<td>erm38</td>
<td>Inducible</td>
<td>Yes</td>
<td>-</td>
<td>AUG</td>
<td>19</td>
<td>62</td>
<td>14506008, 16127056</td>
</tr>
<tr>
<td>erm39</td>
<td>Inducible</td>
<td>No</td>
<td>-</td>
<td>AUG</td>
<td>11</td>
<td>67</td>
<td>15590712</td>
</tr>
<tr>
<td>erm41</td>
<td>Inducible</td>
<td>No</td>
<td>GGAG</td>
<td>AUG</td>
<td>21</td>
<td>55</td>
<td>19171799</td>
</tr>
<tr>
<td>msrA</td>
<td>Inducible</td>
<td>Yes</td>
<td>AAGGAGG</td>
<td>AUG</td>
<td>8</td>
<td>231</td>
<td>2233255</td>
</tr>
<tr>
<td>msrC</td>
<td>Unknown</td>
<td>Yes</td>
<td>AAGGAGGU</td>
<td>AUG</td>
<td>15</td>
<td>226</td>
<td>11120975</td>
</tr>
<tr>
<td>msrD</td>
<td>Inducible</td>
<td>Yes</td>
<td>GGAGG</td>
<td>AUG</td>
<td>6</td>
<td>78</td>
<td>10952626, 11398110, 16223938</td>
</tr>
<tr>
<td>mefA</td>
<td>Unknown</td>
<td>No</td>
<td>AGGAGG</td>
<td>AUG</td>
<td>8</td>
<td>284</td>
<td>11398110</td>
</tr>
<tr>
<td>mefB</td>
<td>Unknown</td>
<td>No</td>
<td>-</td>
<td>AUG</td>
<td>6</td>
<td>168</td>
<td>19131424</td>
</tr>
<tr>
<td>ereA</td>
<td>Unknown</td>
<td>No</td>
<td>GGAG</td>
<td>AUG</td>
<td>15</td>
<td>62</td>
<td>12654734</td>
</tr>
<tr>
<td>ereA</td>
<td>Constitutive</td>
<td>No</td>
<td>GGAG</td>
<td>AUG</td>
<td>16</td>
<td>81</td>
<td>14506050</td>
</tr>
<tr>
<td>ereA</td>
<td>Unknown</td>
<td>No</td>
<td>-</td>
<td>GUG</td>
<td>10</td>
<td>33</td>
<td>12183252</td>
</tr>
<tr>
<td>Gene Name</td>
<td>Inducibility</td>
<td>Annotated leader peptide&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Putative Shine-Dalgaro sequence</td>
<td>Leader ORF start codon</td>
<td>Leader ORF length (amino acid residues)</td>
<td>Distance between leader ORF &amp; resistance gene (bp)</td>
<td>Reference – Pubmed ID</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>---------------------------------</td>
<td>-------------------------------</td>
<td>------------------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td><em>ereA</em></td>
<td>Unknown</td>
<td>No</td>
<td>AAGG</td>
<td>AUG</td>
<td>19</td>
<td>83</td>
<td>12183252</td>
</tr>
<tr>
<td><em>ereB</em></td>
<td>Constitutive</td>
<td>No</td>
<td>GAGG</td>
<td>AUG</td>
<td>21</td>
<td>209</td>
<td>3523438, 2546492</td>
</tr>
<tr>
<td><em>mphA</em></td>
<td>Inducible</td>
<td>No</td>
<td>UAAGG</td>
<td>UUG</td>
<td>17</td>
<td>62</td>
<td>8619599, 10960087</td>
</tr>
<tr>
<td><em>mphB</em></td>
<td>Constitutive</td>
<td>No</td>
<td>GAGG</td>
<td>AUG</td>
<td>11</td>
<td>249</td>
<td>8900063</td>
</tr>
<tr>
<td><em>mphC</em></td>
<td>Unknown</td>
<td>No</td>
<td>AGGU</td>
<td>AUG</td>
<td>8</td>
<td>127</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table footnotes:**

<sup>a</sup> “Yes” indicates that the leader peptide ORF is annotated in the GenBank entry or mentioned in the corresponding reference.

<sup>b</sup> Both inducible and constitutive isolates of this strain were found.

<sup>c</sup> Likely to be constitutive.
D. Identification of putative leader peptides

Our analysis has shown that around 30% of the known macrolide resistance genes had short ORFs in their upstream regions that are good candidates of encoding regulatory peptides. Though most leader genes in the list possess a clearly defined SD sequence, which argues that the respective ORFs are translated thereby regulating the resistance gene by programmed drug-dependent ribosome stalling, we also considered the leader ORFs lacking a SD sequence.

The putative leader peptides having the common motifs namely IFVI, IAVV and RLR were considered to be most promising candidates for further experimental studies to confirm their role in regulation of the downstream gene. An example of such a leader region not previously reported in literature was found in this study in the gene ermQ, a methyltransferase identified from the organism Clostridium perfringens. The potential leader ORF sequence ermQL is MIMNGGIASIRLRR (Table II), has the motif “RLR”.

On the basis of our bioinformatics analysis, a number of the leader ORFs belonging to individual peptide classes were experimentally tested by Dr. Haripriya Ramu and a graduate student Shanmugapriya Sothiselvam in the Mankin lab for the formation of stalled ribosome complex using the technique called toeprinting. It was found that the ‘IFVI’ and ‘IAVV’ type of peptides direct ribosome stalling with the last amino acid of the consensus located in the P-site of the stalled ribosome. The peptides of the RLR class also efficiently stall the ribosome. In this case, the ribosome stalls at the Leu codon. Conservation of the codon (R), following the one at which stalling takes place
suggests the role of the ribosomal A-site in the stalling mechanism. This hypothesis has been recently experimentally confirmed by Shanmugapriya Sothiselvam, who showed that the nature of the A-site codon encoding arginine is critical for the ribosome stalling at the RLR consensus sequence.

Some of the peptides from the miscellaneous group were also tested and shown to direct drug-dependent ribosome stalling. Several of these peptides are currently being experimentally investigated in the Mankin laboratory.
V. CONCLUSION

This study provided a useful starting point for exploring molecular mechanisms of inducible antibiotic resistance mediated by drug- and nascent peptide-dependent ribosome staling. The main outcome of our study is a large collection of leader ORFs in the regulatory regions of macrolide resistance genes, which likely encode regulatory peptides. A variety of the identified putative regulatory peptides provide an unprecedented scope for future experimental exploration of their role in programmed translation arrest. The findings from this project are currently being used by the members of the Mankin lab for studies of mechanisms of ribosome stalling in various antibiotic resistance genes whose leader regions have not been analyzed so far.

The shORF finder proved to be a simple and efficient tool to find possible short ORFs in the upstream region of antibiotic resistance genes. It has been successfully used by several members of the Mankin lab working on various projects. A number of the peptides identified using this program have been subsequently experimentally validated.

Identification of conserved sequence motifs that allowed to group peptides into specific sequence classes provided an additional tool for exploring evolution of the regulatory mechanisms of antibiotic resistance as well as of ribosomal response to the evolutionary-selected stalling peptides.
CITED LITERATURE


APPENDIX

`<!--
---
--
---
---
---
---

# orf.html
---

HTML code for front page
---

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Find short ORFs</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" type="text/css" href="/menus.css"/>
<script language="JavaScript">

function validate()
{
  var str=document.main.sequence.value;
  var str1=document.main.gb.value;
  if(str != "" && str1 != "")
  {
    alert('Please enter any one: either sequence or GenBank ID!');
    return false;
  }
  if(str == "" && str1 == "")
  {
    alert('Please enter sequence or GenBank ID!');
    return false;
  }
  return true;
}

function example()
{
  mySeq = ">gi|2190969|gb|AF002716.1|AF002716 Streptococcus pyogenes leader peptides 1 (lpg1) and 2 (lpg2), and erythromycin resistance methylase (ermTR) genes, comple...
  window.document.main.sequence.value= mySeq; 
  window.document.main.genestart.value= '211';
}

window.document.main.sequence.value= 'AF002716';
window.document.main.genestart.value= '211';
</script>

<body style="margin-left: 0em; margin-top:0em; background-color:#FFFFFF;" dir="ltr">
<table align="center" border="0" cellpadding="0" cellspacing="0" width="778px">
<tr><td>
<map name="uic">
<area shape="rect" coords="0,0,420,100" href="../" alt="lab">
<area shape="rect" coords="504,7,772,31" href="http://www.uic.edu" target="_blank" alt="uic">
<area shape="rect" coords="427,70,773,95" href="http://www.uic.edu/pharmacy/centers/pharmaceutical_biotechnology/" target="_blank" alt="cpb">
</map>
<img src="/images/labpage1.jpg" usemap="#uic" alt="Homepage">
</td></tr>
</body>"
### Identifying short ORFs in a given sequence

Enter GenBank ID or the nucleotide sequence. Also enter the gene start position, if known.

The program identifies all short peptides, of length 4 to 50 amino acids. If gene start position is given, the program searches for short ORFs 500 bases upstream of the given start site. Else, the program searches for all available short ORFs in the given sequence.

Short ORFs with and without Shine-Dalgarno sequence in both the forward as well as reverse frames are displayed.

<table>
<thead>
<tr>
<th><strong>Enter GenBank ID or the nucleotide sequence.</strong> (Example)</th>
<th><strong>Enter gene start position:</strong> (Example)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="example1" alt="" /></td>
<td><img src="example" alt="" /></td>
</tr>
</tbody>
</table>
#!/usr/bin/perl

use strict;
use CGI qw(:standard);
use PerlModules;
use orf_finder;
use w_orf_finder;

print header;
my ($seq, $s, $dna, $gb, $genestart, $dna_sequence, $f1, $f2, $f3, $f4, $f5, $f6) = '';
my ($genbank_id, $accession, $gb_id, $header, $sequence, $organism, $pmid, $fasta_file);

$sequence = param('sequence');
$genbank_id = param('gb');
$genestart = param('genestart');

$dna  = $sequence;
$gb = $genbank_id;

if($genbank_id)
{
  ($sequence) = PerlModules::GetFastaFromGenBankID($genbank_id);
  #print "$sequence<br>
}

if($sequence =~ /^\>\s*/g)
{
  ($gb_id, $header, $seq) = PerlModules::GetSeqFromFasta($sequence);
  ($organism, $pmid, $s) = PerlModules::GetGenBankInfo($gb_id);
} else 
{
  $seq = $sequence;
  $seq =~ s/\[\s\d\]///g;
}

if($seq =~ /[^ATGCNSURYKMWBDHV]/gi)
{
  print <<Back;
    <html>
    <title>Error in the sequence!</title>
    </html>
Back
}
<head>
  <META HTTP-EQUIV="Refresh" CONTENT="1;URL=../orf.html">
</head>
<body onLoad="alert('DNA sequence contains non-standard bases. Please enter a valid DNA sequence only. Page will be refreshed automatically..')">
Back
  exit;
}
elsif($seq eq '')
{
  print <<Back;
  <html>
    <title>Error in GenBank ID!</title>
    <head>
      <META HTTP-EQUIV="Refresh" CONTENT="1;URL=../orf.html">
    </head>
    <body onLoad="alert('Please enter a valid GenBank ID only. Page will be refreshed automatically..')">
Back
  exit;
}

print <<EOF1;
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<title>Possible ORFs in the sequence</title>
<head>
  <link rel="stylesheet" type="text/css" href="/menus.css"/>
  <script language="JavaScript">
    function display()
    {
      document.getElementById('sequence').style.display='block';
      document.getElementById('reset').style.display='block';
    }
    function display1()
    {
      document.getElementById('wsd').style.display='block';
      document.getElementById('withoutsd').style.display='none';
    }
    function display2()
    {
      document.getElementById('wsd').style.display='none';
      document.getElementById('withoutsd').style.display='block';
    }
    function display3()
    {
      document.getElementById('fwd').style.display='block';
      document.getElementById('rev').style.display='none';
    }
    function display4()
    {
      document.getElementById('fwd1').style.display='none';
      document.getElementById('rev1').style.display='block';
    }
    function display5()
    {
      document.getElementById('fwd1').style.display='block';
      document.getElementById('rev1').style.display='none';
    }
    function display6()
    {
      document.getElementById('fwd1').style.display='none';
      document.getElementById('rev1').style.display='block';
    }
    function clear_seq()
    {

document.getElementById('sequence').style.display='none';
document.getElementById('reset').style.display='none';

</script>
</head>
<body style="margin-left: 0em; margin-top:0em; background-color:#FFFFFF;" dir="ltr">
<a name="top"><table align="center" border="0" cellpadding="0" cellspacing="0" width="100%">
<tr><td>
<br><br>
<tr bgcolor="#b1ddff">
<td align="center" height="25"><font face="Arial" color="#000"><b>Results - Short ORFs in the upstream region of gene</b></font></td>
<td align="center"><a href="../orf.html">Find short ORFs</a></td>
</tr>
<tr><td><br><br><br></td></tr>
</table>
<br><br><br>
EOF1

if($gb_id){
    print "<br>Title: &nbsp;&nbsp; $header</b><br>
<br>

EOF1

PerlModules::Print_Seq($seq);
print "<br>Title: &nbsp;&nbsp; $header</b><br>
<br>

EOF2

EOF2

### PerlModules.pm

**Collection of various Perl modules**

```perl
#!/usr/bin/perl

use LWP::Simple;

package PerlModules;

use strict;

sub BaseCount {
    my ($seq) = @_; # Getting the value from the @_ array which stores the DNA sequence passed to subroutine
    my ($a, $t, $g, $c, $n); # Initializing the variables to 0
    $a = 0; $t = 0; $g = 0; $c = 0; $n = 0;
    # Counting each base
    while ($seq =~ /A/g) { $a++ };
    while ($seq =~ /T/g) { $t++ };
    while ($seq =~ /G/g) { $g++ };
    while ($seq =~ /C/g) { $c++ };
    while ($seq =~ /N/g) { $n++ };
    # ^ operator inside the [] checks for any base or character other than ATGC
    return ($a, $t, $g, $c, $n); # Returning values to the main function
}

sub CheckBase {
    my ($seq) = @_; # Getting the value from the @_ array which stores the DNA sequence passed to subroutine
    # Check if bases other than A, T, G, C and N are found and exit after printing error message
    die "DNA sequence contains non-standard bases. Please enter a valid DNA sequence only.\n\n" if ($seq =~ /[^ATGCNSURYKMWBDHV]/g);
}

sub CheckAminoAcids {
    # Subroutine to check if the bases in the given protein sequence are standard amino acids
}
```

---

**Display ORFs** (Click on the button)

- With Shine-Dalgarino:
  - `<input type="button" name="shine" id="shine" value="With Shine-Dalgarino" onClick="display1()">`
- Including ORFs Without Shine-Dalgarino:
  - `<input type="button" name="wshine" id="wshine" value="Including ORFs Without Shine-Dalgarino" onClick="display2()">`

```perl
EOF3
orf_finder::orf($seq,$genestart);
print <<EOF4;

</td></tr></table>
</div>
```
sub CheckProtein
{
    my ($seq) = @_;  # Getting the value from the @_ array which
    # stores the protein sequence passed to subroutine

    # Check if amino acids other than ARNDCEQGHILKMFPSTWYV and ambiguous codes BEJX
    # and stop codon * are found and exit after printing error message
    die "DNA sequence contains non-standard bases. Please enter a valid DNA sequence
    only.\n\n" if ($seq =~ /[^ARNDCEQGHILKMFPSTWYVBZJX\*/g);
}

### Subroutine to remove numbers and spaces from a sequence
sub RawSeq
{
    my ($seq) = @_;  
    #print "$seq"
    $seq =~ s/\[[s\d]+//g;
    #print "$seq"
    return ($seq);
}

### Subroutine to extract GenBank id and sequence from a given FASTA sequence (from
GenBank)
sub GetSeqFromFasta
{
    my ($seq) = @_;  
    my ($gb_id, $header, $sequence);
    if($seq =~ /^>(gi|\d+|S+|\S+|.*)?\n\n(.*)/sg)
    {
        $header = $1;
        $gb_id = $2;
        $sequence = $3;
    }
    elsif($seq =~ /^>(.*)\n\n(.*)/sg)
    {
        $header = $1;
        $sequence = $2;
    }
    $sequence =~ s/\[[s\d]+//g;
    #print "$seq\n$gb_id\n$sequence\n";
    return ($gb_id, $header, $sequence);
}

sub GetGenBankInfo
{
    my ($gb_id)= @_;  
    my ($url, $sequence, $esearch_result, $organism, $pmid);
    my $db = "nucleotide";
    # $url = "esearch.fcgi?db=$db&usehistory=y&term=$gb_id";
    # $esearch = "$utils$url";
    # $esearch_result = get($esearch);
    #print "** $esearch_result<br><br>";
    # $esearch_result =~ m|<Count>(\d+)</Count>.*<QueryKey>(\d+)/QueryKey.*<WebEnv>(\S+)/WebEnv>|s;
    #my $Count    = $1;
    #my $QueryKey = $2;
    #my $WebEnv   = $3;
    #print "Count = $Count; QueryKey = $QueryKey; WebEnv = $WebEnv\n"
    #my $efetch = "$utils/efetch.fcgi?db=$db&query_key=$QueryKey&WebEnv=$WebEnv";
    #my $efetch = "$utils/efetch.fcgi?db=$db&query_key=$QueryKey&WebEnv=$WebEnv";
    #my $efetch_result = get($efetch);
    #print "\nEF_QUERY=$efetch\n";
    #my $efetch_result = get($efetch);
    #print "$efetch_result"
    return $efetch_result;
}
if($efetch_result eq "")
{
    $db = "nuccore";
    $efetch = "$utils/efetch.fcgi?" . "rettype=gb&retmode=text" .
    "db=$db&id=$gb_id";
    $efetch_result = get($efetch);
}

my $gb_file = "/web_temp/fa/gb.txt";
WriteToFile($efetch_result, $gb_file);
($organism, $pmid, $sequence) = ExtractFromGenBank($gb_file);
system("rm $gb_file");
return ($organism, $pmid, $sequence);

### Subroutine to extract features from GenBank file
sub ExtractFromGenBank
{
    my ($filename, $line, $sequence, $organism, $pmid) = '';
    my (@contents) = ();
    my $flag = 0;

    #print "Enter the name of the GenBank file:
    #chomp ($filename= <STDIN>);
    ($filename) = @_;

    ### Opening the file and reading the data ###
    if(!open (INFILE, "$filename")) { die "Cannot open file $filename";
    @contents = <INFILE>;
    close INFILE;

    foreach $line (@contents)
    {
        if($line =~ /^\s+ORGANISM\s+(.*)/) { $organism = $1; }
        if($line =~ /^\s+PUBMED\s+(.*)/) { $pmid = $1; }
        if($line =~ /^ORIGIN/) # Check for line beginning with ORIGIN
        { $flag = 1; # Activate the flag
            next; # Goto next line
        } elsif ($line =~ /^\s+//) # Check for line with // - End of sequence -
            Break out of the loop
        { $flag = 0; #Deactivate the flag
            last;
        }

        if($flag == 1)
        { $sequence .= $line; # Concatenate rest of lines with the
            sequence variable when flag remains activated
        }
    }

    $sequence =~ s/\s+/\s+//g; # Remove numbers and spaces from the
    $sequence = uc $sequence;
    #print "$organism $pmid $sequence<br>";
    #PrintSeq($sequence,60);
    return ($organism, $pmid, $sequence);
}

### Subroutine to get Fasta sequence given Genbank ID
sub GetFastaFromGenBankID
{
my ($gb_id) = @_;  # my ($url, $content, $sequence);  # my $db = "nucleotide";
$content = get($url);
if($content eq "") {
  $db = "nuccore";
$content = get($url);
}
$fasta_file = "/web_temp/fa/fasta.txt";
WriteToFile($content, $fasta_file);
if(!open (FASTAFILE, "$fasta_file")) { die "Cannot open file $fasta_file\n"; }
my @contents = <FASTAFILE>;
close FASTAFILE;
system ("rm $fasta_file");
$sequence = join('', @contents);
return ($sequence);

### Subroutine to return the reverse complement of the given DNA sequence
sub ReverseComplement {
  my ($dna_seq) = @_;  # my ($rev_comp_dna) = '';  # $dna_seq = uc $dna_seq;  # Convert the sequence into uppercase
  for easier manipulations
  $rev_comp_dna = reverse ($dna_seq);  # Using reverse function, find the reverse of the string
  $rev_comp_dna =~ tr/ATGC/TACG/;  # Translate the bases A to T, T to A, G to C and C to G using "tr" translation operation
  return ($rev_comp_dna);  # Returning reverse complement of DNA
}

### Subroutine to write a given scalar of text to a file specified by the user
sub WriteToFile {
  my ($input_string, $filename) = @_;  # Getting the value from the @_ array which stores the input string
  my ($filename);
  #print "Enter the name of the file to be written into:\n";
  #chomp ($filename = <STDIN>);  # Remove newline and store the filename entered by user in the variable
  if(!open (OUTFILE, ">$filename")) {
    die "Cannot open file $filename\n";  # Using open statement to open the file with name $filename in write mode (">") and assigning a filehandle OUTFILE to this file
  }
  printf OUTFILE "%s\n", $input_string;  # Printing formatted result to the output file using printf
  close OUTFILE;  # Close the file
### Subroutine to get ORF for a given DNA sequence and frame start position

sub OpenReadingFrames
{
    my ($dna_sequence, $DNA, $dna_length, $dna) = '';
    my ($frame_start, $frame_end, $codon) = '';
    ($dna_sequence, $frame_start) = @_;  # Get the DNA sequence and frame start position
    $frame_end = length $dna_sequence;  # Frame length
    $dna = substr($dna_sequence, ($frame_start - 1), ($frame_end - $frame_start + 1));  # Extracting the ORF from frame start to correct length of ORF
    #print "$frame_start<br>
    #print *** $dna_sequence
    #print $dna
    return $dna;
}

### Subroutine to convert a DNA sequence into protein and return the protein sequence

sub Translate
{
    my ($dna_sequence) = @_;  # Declare variables to hold the DNA sequence and converted protein sequence
    my ($protein, $dna_length) = '';
    $dna_sequence = uc $dna_sequence;  # length function determines the length of the string
    for (my $i=0; $i < ($dna_length - 2); $i += 3)  # For every triplet of bases, we run this loop and stop when there are less than 2 bases
    {
        $codon = substr($dna_sequence, $i, 3);  # substr cuts a subsequence from position specified by $i upto 3 positions from the string $dna
        $aa = CodonToAA($codon);  # Send this substring which is the codon to the subroutine and concatenate the results, which is the corresponding aminoacid into the $protein string using the .= operator
        $protein .= $aa;
    }
    return $protein;  # Return the protein sequence
}

### Subroutine to return the amino acid corresponding to a codon

sub CodonToAA
{
    ### Declaration of variables ###
    my ($codon, %codon_hash);  # Declaring a scalar variable to store the codon and a hash variable with key-value pair;
    # The hash holds the codon as key since it is unique and the corresponding 1-letter amino acid code as the value.
    ($codon) = @_;  # Get the codon from the @_ array which stores the variables sent from the subroutine call
    %codon_hash = ();
    $codon =~ s/U/T/g;  # If there is U in the codon, translate that to T for uniformity in applying the genetic code
    #print "$codon\n";

    ### Populating the hash variable with codons and amino acid single letter codes
    %codon_hash = {
        'TTT' => 'F',  # Codes for AA Phenylalanine
        'TTC' => 'F',  # Codes for AA Phenylalanine
        'TTA' => 'L',  # Codes for AA Leucine
        'TTG' => 'L',  # Codes for AA Leucine
        'CTT' => 'L',  # Codes for AA Leucine
        'CTC' => 'L',  # Codes for AA Leucine
    };
}
'CTA'  => 'L',  # Codes for AA Leucine
'CTG'  => 'L',  # Codes for AA Leucine
'ATT'  => 'I',  # Codes for AA Isoleucine
'ATC'  => 'I',  # Codes for AA Isoleucine
'ATG'  => 'M',  # Codes for AA Methionine
'GTT'  => 'V',  # Codes for AA Valine
'GTC'  => 'V',  # Codes for AA Valine
'GTA'  => 'V',  # Codes for AA Valine
'CTC'  => 'V',  # Codes for AA Valine
'TCT'  => 'S',  # Codes for AA Serine
'TCC'  => 'S',  # Codes for AA Serine
'TCA'  => 'S',  # Codes for AA Serine
'TCG'  => 'S',  # Codes for AA Serine
'CCT'  => 'P',  # Codes for AA Proline
'CCC'  => 'P',  # Codes for AA Proline
'CCA'  => 'P',  # Codes for AA Proline
'CCG'  => 'P',  # Codes for AA Proline
'ACT'  => 'T',  # Codes for AA Threonine
'ACC'  => 'T',  # Codes for AA Threonine
'ACA'  => 'T',  # Codes for AA Threonine
'ACG'  => 'T',  # Codes for AA Threonine
'GCT'  => 'A',  # Codes for AA Alanine
'GCC'  => 'A',  # Codes for AA Alanine
'GCA'  => 'A',  # Codes for AA Alanine
'GCG'  => 'A',  # Codes for AA Alanine
'TAT'  => 'Y',  # Codes for AA Tyrosine
'TAC'  => 'Y',  # Codes for AA Tyrosine
'TAA'  => '*',  # Codes for Stop codon
'TAG'  => '*',  # Codes for Stop codon
'CAT'  => 'H',  # Codes for AA Histidine
'CAC'  => 'H',  # Codes for AA Histidine
'CAA'  => 'Q',  # Codes for AA Glutamine
'CAG'  => 'Q',  # Codes for AA Glutamine
'AAT'  => 'N',  # Codes for AA Asparagine
'AAC'  => 'N',  # Codes for AA Asparagine
'AAA'  => 'K',  # Codes for AA Lysine
'AAG'  => 'K',  # Codes for AA Lysine
'GAT'  => 'D',  # Codes for AA Aspartic acid
'GAC'  => 'D',  # Codes for AA Aspartic acid
'GAA'  => 'E',  # Codes for AA Glutamic acid
'GAG'  => 'E',  # Codes for AA Glutamic acid
'TGT'  => 'C',  # Codes for AA Cysteine
'TGC'  => 'C',  # Codes for AA Cysteine
'TGA'  => '*',  # Codes for Stop codon
'TGG'  => 'W',  # Codes for AA Tryptophan
'CCT'  => 'R',  # Codes for AA Arginine
'CCT'  => 'R',  # Codes for AA Arginine
'CGA'  => 'R',  # Codes for AA Arginine
'CGG'  => 'R',  # Codes for AA Arginine
'AGT'  => 'S',  # Codes for AA Serine
'AGC'  => 'S',  # Codes for AA Serine
'AGA'  => 'R',  # Codes for AA Arginine
'AGG'  => 'R',  # Codes for AA Arginine
'GGT'  => 'G',  # Codes for AA Glycine
'GGC'  => 'G',  # Codes for AA Glycine
'GGA'  => 'G',  # Codes for AA Glycine
'GGG'  => 'G',  # Codes for AA Glycine

);  # Check if the codon exists in the hash and return the hash value (amino acid) for the codon key
if (exists $codon_hash{$codon})
{
  return $codon_hash{$codon};
}
elsif($codon =~ /[.N|.N.|N..]/)
{
  $codon_hash{$codon} = "X";
}
else
{
  die "Codon does not exist $codon! Please try again.\n";
### Subroutine for printing specified number of characters per line

```perl
sub PrintSeq {
    my ($seq, $number_per_line) = @_;  # Get the sequence and number of characters to be printed in every line
    if ($number_per_line == '') { $number_per_line = "60"; }
    my $seq_length = length($seq);
    for (my $i=0; $i < $seq_length; $i += $number_per_line) {
        my $seq_print = substr($seq, $i, $number_per_line);  # Extracting the number of characters to be printed per line
        print "$seq_print
";  # Printing the extracted number of characters in one line
    }
}
```

### Subroutine to extract DNA sequence from GenBank file

```perl
sub ExtractDNAFromGenBank {
    my ($filename, $line, $sequence) = '';  
    my (@contents) = ();
    my $flag = 0;
    #print "Enter the name of the GenBank file:
";
    #chomp ($filename= <STDIN>);
    ($filename) = @_;  
    ### Opening the file and reading the data ###
    if(!open (INFILE, "$filename")) {
        die "Cannot open file $filename
"
    }
    @contents = <INFILE>;
    close INFILE;
    foreach $line (@contents) {
        if($line =~ /^ORIGIN/)  # Check for line beginning with ORIGIN
            {
                $flag = 1;  # Activate the flag
                next;  # Goto next line
            }
        elsif ($line =~ ^\s*$/)  # Check for line with // - End of sequence -
            break out of the loop
            {
                $flag = 0;  # Deactivate the flag
                last;
            }
        if($flag == 1)
            {
                $sequence .= $line;  # Concatenate rest of lines with the sequence variable when flag remains activated
            }
    }
    $sequence =~ s/\d+/\d*/g;  # Remove numbers and spaces from the sequence
    #PrintSeq($sequence,60);  # Returning extracted sequence
    return ($sequence);
}
```

### Subroutine to extract protein sequence from translation feature of GenBank file

```perl
sub ExtractProteinFromGenBank {
    ...
}
my ($filename, $line, $sequence) = '';  
my (@contents) = ();  
my $flag = 0;  

# Print "Enter the name of the GenBank file:\n"; 
# chomp ($filename=<STDIN>); 

($filename) = {};  

### Opening the file and reading the data ###  
if(!open (INFILE, "$filename"))  
{ 
  die "Cannot open file $filename\n"; 
}  

@contents = <INFILE>;  
close INFILE;  

foreach $line (@contents)  
{  
  if($line =~ /
translation=\n/)  
    # Check for line beginning with translation  
    {
      $flag = 1;  
      # Activate the flag - Extract from same line  
    }
  elsif($line =~ /^ORIGIN/)  
    # Check for line beginning with ORIGIN - 
    where protein translation ends  
    {
      $flag = 0;  
      # Deactivate the flag  
      last;  
    }
  if($flag == 1)  
  {  
    $sequence .= $line;  
    # Concatenate rest of lines with the sequence variable when flag remains activated  
  }

$sequence =~ s/\ntranslation=\n//g;  
$sequence =~ s/\s\s//g;  
# Remove spaces and " from the sequence  
# PrintSeq($sequence, 60);  
return ($sequence);  
# Returning extracted sequence

### Subroutine for printing specified number of characters per line (HTML) ###  
sub Print_Seq  
{
  my ($seq) = {};  
  # Get the sequence to be printed in every line  
  my ($send_count, $seq_print, $number_per_line, $seq_length, $count, $diff, $i, $j, $seq_part);  
  $number_per_line = "80";  
  $seq_length = length ($seq);  
  $count = 1;  
  print "<pre>";  

  $diff = ($seq_length % $number_per_line);  
  print "$seq_length\nt*";  

  for ($i=0; $i < $seq_length; $i += $number_per_line)  
  {
    $seq_print = substr ($seq, $i, $number_per_line);  
    # Extracting the number of characters to be printed per line  
    print "$count\t";
    if($count != ($seq_length - $diff + 1))  
    {  
      $send_count = $count + $number_per_line - 1;  
    }
    else  
    {  
      $send_count = $count + $diff - 1;  
    }

"
for($j=0; $j < length $seq_print ; $j += 10)
{
    $seq_part = substr($seq_print, $j, 10);
    print "$seq_part "; # Printing the extracted number of
characters in one line
}
print"<t$end_count<br>";
$count += $number_per_line;
}
print "</pre>";
($seq, $seq_print, $count, $end_count, $diff) = '';
1;

##########################################################
##############################
#### orf_finder.pm #### Find short ORFs with SD sequence ####
#!/usr/bin/perl
# orf_finder.pm
package orf_finder;

use strict;
use CGI qw(:standard);
use PerlModules;

sub orf {
    my ($seq, $sequences, $s, $dna_sequence, $f1, $f2, $f3, $f4, $f5, $f6) = ''; 
    my ($genestart, $ups_start, $upstream, $ups_start_rev, $rev_seq);
    my ($genbank_id, $accesion, $gb_id, $header, $sequence, $organism, $pmid, $fasta_file);

    $sequences = $_[0];
    $genestart = $_[1];

    my $length_sequences = length $sequences;
    if($genestart)
    {
        $ups_start = $genestart - 500;
        $ups_start_rev = $genestart;
        if($ups_start > 0)
        {
            $seq = substr($sequences, $ups_start, 500);
        } else
        {
            $ups_start = 0;
            $seq = substr($sequences, 0, $genestart - 1);
        }
        if($length_sequences - $genestart > 0)
        {
            $rev_seq = substr($sequences, $genestart, 500);
        } else
        {
            $rev_seq = $sequences;
        }
    } else
    {
        $genestart = 0;
        $seq = $sequences;
        $rev_seq = $sequences;
    }
    print <<EOF3;
    <table border="0" cellpadding="2" cellspacing="2" align="center" width="100%">
    <tr>
Possible short ORFs in the sequence - With Shine-Dalgarno (SD)

### ORF 1
my $dna_sequence = PerlModules::OpenReadingFrames($original_dna,1);
$f1 = leader($dna_sequence,$seq,$ups_start,$genestart,1);

### ORF 2
my $dna_sequence2 = PerlModules::OpenReadingFrames($original_dna,2);
$f2 = leader($dna_sequence2,$seq,$ups_start,$genestart,2);

### ORF 3
my $dna_sequence3 = PerlModules::OpenReadingFrames($original_dna,3);
$f3 = leader($dna_sequence3,$seq,$ups_start,$genestart,3);

print "EOF10;
</div>
<div id="rev" style='display: none;'>
<ul>
<li><a href="#Reverse 1">Reverse 1</a>
<li><a href="#Reverse 2">Reverse 2</a>
<li><a href="#Reverse 3">Reverse 3</a>
</ul>
EOF10

### Reverse direction
my $rev_comp_dna = reverse $rev_seq;
$rev_comp_dna =~ tr/ATGC/TACG/;

my $original_rev_dna = $rev_comp_dna;

### ORF 1
my $dna_sequence4 = PerlModules::OpenReadingFrames($original_rev_dna,1);
$f4 = leader($dna_sequence4,$rev_comp_dna,$ups_start_rev,$genestart,4);

### ORF 2
my $dna_sequence5 = PerlModules::OpenReadingFrames($original_rev_dna,2);
$f5 = leader($dna_sequence5,$rev_comp_dna,$ups_start_rev,$genestart,5);

### ORF 3
my $dna_sequence6 = PerlModules::OpenReadingFrames($original_rev_dna,3);
$f6 = leader($dna_sequence6,$rev_comp_dna,$ups_start_rev,$genestart,6);

print "</div>";

if (($f1 == 0) && ($f2 == 0) && ($f3 == 0) && ($f4 == 0) && ($f5 == 0) && ($f6 == 0)) {
    print "<p align="center"><font face="Arial" size="4" color="Red">Maybe there is no ORF in the given sequence....</p></font>";
}

($seq, $rev_comp_dna, $f1, $f2, $f3, $f4, $f5, $f6) = '';
Subroutines

Subroutine - Find start codon

```perl
my $flag = 0; my $f = 0;
my($seq, $dna, $ups_start, $genestart, $orf) = @_; # $seq - Frameshifted DNA;
$dna - Original sequence

my ($strand, $seq_length, $m, $codon, $off, $dna_length, $start);

if($orf eq "1") { $strand = "Forward 1"; $off = 0; }
elsif ($orf eq "2") { $strand = "Forward 2"; $off = 1; }
elsif ($orf eq "3") { $strand = "Forward 3"; $off = 2; }
elsif ($orf eq "4") { $strand = "Reverse 1"; $off = 0; }
elsif ($orf eq "5") { $strand = "Reverse 2"; $off = 1; }
elsif ($orf eq "6") { $strand = "Reverse 3"; $off = 2; }

$dna_length = length $dna;
$seq_length = length $seq;

print "$seq<br>

$co = 0;
print "<a name="$strand">\n<p align="left"><font face="Arial" size="4" color="Red">\n<b>$strand</b></font>\n</p></a></font><br><br>
<EOF1>
<table border="1" cellpadding="8" cellspacing="4" width="776px">
<tr align="center" bgcolor="#b1ddff">
<th>Sequence upstream of start codon<br>
<th>SD position<br>
<th>SD sequence<br>
<th>Spacer (length)<br>
<th>Start codon<br>
<th>ORF position<br>
<th>ORF length<br>
<th>Peptide length<br>
<th>Distance of shORF from gene<br>
<th>Nucleotide sequence of ORF<br>
<th>Peptide sequence<br>
</tr>

EOF1

for($m = 0; $m <= $seq_length; $m += 3) {
    $codon = substr($seq,$m,3);
    #print "$codon\n";
    if($codon =~ /ATG/) # Check if codon is ATG - any occurrence
        {
            $flag = FindLP($m, $codon, $seq, $off, $ups_start, $genestart,
            $orf);
        } # if $codon loop
    elsif($codon =~ /GTG/) # Check if codon is ATG - any occurrence
        {
            $flag = FindLP($m, $codon, $seq, $off, $ups_start, $genestart,
            $orf);
        }
    elsif($codon =~ /TTG/) # Check if codon is ATG - any occurrence
        {
            $flag = FindLP($m, $codon, $seq, $off, $ups_start, $genestart,
            $orf);
        }
    $f = $f + $flag;
    #print "+++++++ m = $m ==== co = $co *** LP end $lp_end ++++ <br><br>
} # for loop

print "Flag $f<br>";
if($f == 0) {

```
print "<tr align="center">
<td colspan=11>There is no peptide in this frame</td></tr>
";
}

print "</table><br><a href="#top">Top</a>";
return $f;
} # subroutine end

sub FindLP {
my $flag = 0;
my ($orf, $dna_length, $ups_start, $genestart, $seq_length, $co, $m, $codon, $off, $seq);  
my ($final_seq, $total_length, $cod, $subseq_start, $subseq);  
my ($lp_start, $lp_seq, $lp_end, $peptide, $lp_length, $peptide_len, $lp_start_codon, $start);  
my ($shine, $sd_start, $sd_length, $sd_end, $spacer, $spacer_length);
($m, $codon, $seq, $off, $ups_start, $genestart, $orf) = @_;  
# print "$m $codon $start"
$start = $codon;
$lp_start = $m;

$seq_length = length $seq;
$co = 0;
# print "LP start $lp_start *** Start codon $codon <br><br>";
$lp_seq .= $codon; # ATG appended to LP sequence
for($co = $lp_start + 3; $co < $seq_length; $co += 3) {
$cod = substr($seq, $co, 3);
if($cod !~ /(TAA|TGA|TAG)/) {
    $lp_seq .= $cod;
} else {
    $lp_seq .= $cod;
    last;
}
# print "<br> &&&& $ lp_seq & & & <br> <br>";
} my $lp_fragment = substr($lp_seq, 0, 9);
# print "$lp_fragment<br>";
$peptide = PerlModules::Translate($lp_seq);
$peptide =~ s/^[L|V]/M/;
# print "<br>";
$lp_length = length $lp_seq; # LP Nucleotide seq length
$lp_end = $lp_start + $lp_length - 1;
$peptide_len = length $peptide; # peptide length
my $last_aa = substr($peptide, $peptide_len - 1, 1);
# print "$last_aa <br>";
if($peptide ne "" && $peptide_len >= 4 && $peptide_len <= 50 && $last_aa eq "*") # Peptide not empty and has a stop codon {
    $subseq_start = $lp_start - 30;
    if($subseq_start > 0) {
        $subseq = substr($seq, $subseq_start, 39);
    } else
$subseq = substr($seq, 0, $lp_start + 9);
}

# print "<br>$subseq</b><br>";

my (@sd, @len, @arr, @a, @unique_subseq, $sd_seq);
#
$sd = ('TAAGGAGGT', 'AAAGAGGTG');
$sd = ('TAAGGAGGT');

foreach (@sd) {
  $len = length $_;
  for(my $i=4; $i <= $len; $i += 1) {
    for(my $j=0; $j <= ($len - $i); $j += 1) {
      # print "$i<\t$j<br>"
      my $seq = substr($_, $j, $i);
      @{$arr{$seq}} = 1;
    }
  }
}

@a = keys @arr;
@unique_subseq = sort {length $b <=> length $a || $b cmp $a} @a;

foreach $sd_seq (@unique_subseq) {
  if($subseq =~ /($sd_seq)(.{3,20})($lp_fragment)/) {
    $shine = $1;
    $spacer = $2;
    $lp_start_codon = $start;
    $sd_length = length $shine;
    $spacer_length = length $spacer;
    $sd_end = $sd_start + $sd_length - $spacer_length - 1;
    # print "DNA ups_start ### Frame $seq_length SD start $sd_start SD end $sd_end LP start $lp_start end $lp_end<br>"
    if($orf == "4" || $orf == "5" || $orf == "6") {
      $sd_end = $seq_length - $sd_end + $ups_start;
      $lp_end = $seq_length - $lp_end + $ups_start;
      $lp_start = $lp_end - $sd_end + $ups_start;
      $lp_start = $lp_start - $off + $ups_start;
      $lp_end = $lp_end - $off + $ups_start;
    } else {
      $sd_start = $sd_start + 1 + $off + $ups_start;
      $sd_end = $sd_end + 1 + $off + $ups_start;
      $lp_start = $lp_start + 1 + $off + $ups_start;
      $lp_end = $lp_end + 1 + $off + $ups_start;
    }
    my $distance = $genestart - $lp_end - 1;
    if($orf == "4" || $orf == "5" || $orf == "6") {
      $distance = $lp_end - $genestart - 1;
    } if($distance <= 0) {$distance = ";"; }
    $subseq =~ s/T/U/g;
    $shine =~ s/T/U/g;
    $spacer =~ s/T/U/g;
    $lp_start_codon =~ s/T/U/g;
    $lp_seq =~ s/T/U/g;
}

$peptide_len = $peptide_len - 1;
print "<tr align="center"><n<td>
PerlModules::PrintSeq($subseq,10);
print "\n<td>$sd_start - $sd_end\n</td><font

color="\red\">$shine</font>\n</td>
PerlModules::PrintSeq($spacer,7);
print "\nbsp;\nbsp; ($spacer_length)\n";
print "\n</td>$lp_start_codon</n<td>
PerlModules::PrintSeq($lp_seq,30);
print "$peptide
</tr>
$flag++;($lp_seq, $peptide, $lp_start, $lp_end, $sd_start, $sd_end, $subseq_start, $subseq) = '';
if($length_sequences - $genestart > 0)
{
    $rev_seq = substr($sequences, $genestart, 500);
}
else
{
    $rev_seq = $sequences;
}
else
{
    $genestart = 0;
    $seq = $sequences;
    $rev_seq = $sequences;
}

print <<EOF3;
<table border="0" cellpadding="2" cellspacing="2" align="center" width="100%">
<tr>
    <td bgcolor="#b1ddff" align="center" height="25">
        <font face="Arial" color="#000"><b>Possible short ORFs in the sequence - Without Shine-Dalgarno (SD)</b></font>
    </td>
</tr>
</table>
<p><font size="4" color="Red">Strand</font> &nbsp;&nbsp;
<input type="button" name="forward1" id="forward1" value="Forward" onclick="display5()">
<input type="button" name="reverse1" id="reverse1" value="Reverse" onclick="display6()"></input>
<div id="fwd1" style='display: block;'><ul>
    <li><a href="#Forward 1a">Forward 1</a>
    <li><a href="#Forward 2a">Forward 2</a>
    <li><a href="#Forward 3a">Forward 3</a>
</ul></div>
EOF3

my $original_dna = $seq;

### ORF 1
$seq = PerlModules::OpenReadingFrames($original_dna,1);
$f1 = leader1($seq,$seq,$ups_start,$genestart,1);

### ORF 2
my $dna_sequence2 = PerlModules::OpenReadingFrames($original_dna,2);
$f2 = leader1($dna_sequence2,$seq,$ups_start,$genestart,2);

### ORF 3
my $dna_sequence3 = PerlModules::OpenReadingFrames($original_dna,3);
$f3 = leader1($dna_sequence3,$seq,$ups_start,$genestart,3);

print <<EOF10;
</div>
<div id="rev1" style='display: none;'><ul>
    <li><a href="#Reverse 1a">Reverse 1</a>
    <li><a href="#Reverse 2a">Reverse 2</a>
    <li><a href="#Reverse 3a">Reverse 3</a>
</ul></div>
EOF10

### Reverse direction
my $original_rev_dna = reverse $rev_seq;
$rev_comp_dna =~ tr/ATGC/TACG/;

my $original_rev_dna = $rev_comp_dna;

### ORF 1
my $dna_sequence4 = PerlModules::OpenReadingFrames($original_rev_dna,1);
$f4 = leader1($dna_sequence4,$rev_comp_dna,$ups_start_rev,$genestart,4);
### ORF 2
my $dna_sequence5 = PerlModules::OpenReadingFrames($original_rev_dna,2);
$f5 = leader1($dna_sequence5,$rev_comp_dna,$ups_start_rev,$genestart,5);

### ORF 3
my $dna_sequence6 = PerlModules::OpenReadingFrames($original_rev_dna,3);
$f6 = leader1($dna_sequence6,$rev_comp_dna,$ups_start_rev,$genestart,6);

print "</div>");
if (($f1 == 0) && ($f2 == 0) && ($f3 == 0) && ($f4 == 0) && ($f5 == 0) && ($f6 == 0))
{
  print "<p align="center"><font face="Arial" size="4" color="Red"><br>Maybe there is no ORF in the given sequence...<br><br></font></p>";
}
($seq, $rev_comp_dna, $f1, $f2, $f3, $f4, $f5, $f6) = ''; ($dna_sequence, $dna_sequence2, $dna_sequence3, $dna_sequence4, $dna_sequence5, $dna_sequence6) = '';
if($codon =~ /ATG/)  # Check if codon is ATG - any occurrence
   { $flag = FindLP1($m, $codon, $seq, $off, $ups_start, $genestart, $orf); }
   # if $codon loop
elsif($codon =~ /GTG/)  # Check if codon is ATG - any occurrence
   { $flag = FindLP1($m, $codon, $seq, $off, $ups_start, $genestart, $orf); }
elsif($codon =~ /TTG/)  # Check if codon is ATG - any occurrence
   { $flag = FindLP1($m, $codon, $seq, $off, $ups_start, $genestart, $orf); }
$f = $f + $flag;
   #print "+++++++ m = $m ==== co = $co *** LP end $lp_end ++++ <br><br>";
} # for loop
#print "Flag $f<br>";
if($f == 0)
   {
      print "<tr align="center"><td colspan=9>There is no peptide in this frame</td></tr>";
   }
   print "</table><br><a href="#top">Top</a><hr>";
return $f;
} # subroutine end
sub FindLP1
{
   my $flag = 0; my $shd = 0;
   my ($orf, $dna_length, $ups_start, $genestart, $seq_length, $co, $m, $codon, $off, $seq);
   my ($final_seq, $total_length, $cod, $subseq_start, $subseq);
   my ($lp_start, $lp_seq, $lp_end, $peptide, $lp_length, $peptide_len, $lp_start_codon, $start);
   my ($shine, $sd_start, $sd_length, $sd_end, $spacer, $spacer_length);
   ($m, $codon, $seq, $off, $ups_start, $genestart, $orf) = @_;  
   #print "$m\t$codon\t$start\n<br>";
   $start = $codon;  
   $lp_start = $m;  
   $seq_length = length $seq;  
   $co = 0;  
   #print "LP start $lp_start *** Start codon $codon <br><br>";
   $lp_seq .= $codon;  
   # ATG appended to LP sequence
   for($co = $lp_start + 3; $co < $seq_length; $co += 3)  # Extract nts after start codon
      {
      $cod = substr($seq,$co,3);  
      #print "$co *** $cod\n";
      if($cod !~ /TAA|TGA|TAG/)
      { $lp_seq .= $cod;  
      } else
      { $lp_seq .= $cod;  
      last);
      }  
   #print "<br> &&& $lp_seq &&& <br><br>
   }
my $lp_fragment = substr($lp_seq, 0, 9);
$peptide = PerlModules::Translate($lp_seq);
$peptide =~ s/^[L|V]/M/;

# print "<br>
$lp_length = length $lp_seq; # LP Nucleotide seq length
$lp_end = $lp_start + $lp_length - 1;

$peptide_len = length $peptide; # peptide length
my ($last_aa) = substr($peptide, $peptide_len - 1, 1);
if($peptide ne "" && $peptide_len >= 4 && $peptide_len <= 50 && $last_aa eq "+")
# Peptide not empty and has a stop codon
{
  $subseq_start = $lp_start - 30;
  if($subseq_start > 0)
  {
    $subseq = substr($seq, $subseq_start, 39);
  }
  else
  {
    $subseq = substr($seq, 0, $lp_start + 9);
  }
  #print "<br><b>$subseq</b><br>

my ($sd, $len, %arr, @a, %unique_subseq, $sd_seq);
$sd = ("TAAGGAGGT", "AAAGAGGTG");
$sd = ("TAAGGAGGT");

foreach (@sd)
{
  $len = length $_;
  for(my $i=4; $i <= $len; $i += 1)
  {
    for(my $j=0; $j <= ($len - $i); $j += 1)
    {
      my $subseq = substr($_, $j, $i);
      $arr{$subseq} = 1;
    }
  }
}
@a = keys %arr;
@unique_subseq = sort {length $b <=> length $a || $b cmp $a} @a;
foreach $sd_seq (@unique_subseq)
{
  if($subseq =~ /$sd_seq.{3,20}$lp_fragment/) {$shd = "Yes";}
}
if($shd ne "Yes") {$shd = "No";}
if($orf == "4" || $orf == "5" || $orf == "6")
{
  $lp_end = $seq_length - $lp_end + $ups_start;
  $lp_start = $lp_end + $lp_length - 1;
}
else
{
  $lp_start = $lp_start + 1 + $off + $ups_start;
  $lp_end = $lp_end + 1 + $off + $ups_start;
}

my $distance = $genestart - $lp_end - 1;
if($orf == "4" || $orf == "5" || $orf == "6")
{
  $distance = $lp_end - $genestart - 1;
}
if($distance <= 0) ($distance = "+")
$subseq =~ s/T/U/g;
$start =~ s/T/U/g;
$lp_seq =~ s/T/U/g;
$peptide_len = $peptide_len - 1;

print "<tr align="center">
\n<td>
PerlModules::PrintSeq($subseq,10);

print "<td>$shd\n";
print "<td><font color="#00ff00"><b>$start</b></font>\n<td>$lp_start - $lp_end\n<td>$peptide_len\n";
print "<td>$distance\n";

if($lp_length <= 30)
{
  print "$lp_seq\n</tr>
}
else
{
  PerlModules::PrintSeq($lp_seq,30);
  print "\n</td>";
  PerlModules::PrintSeq($peptide,30);
  print "</tr>";
}
$flag++;

($lp_seq, $peptide, $lp_start, $lp_end, $subseq_start, $subseq) = '';
VITA

NAME: Sai Lakshmi Subramanian

EDUCATION: B. E., Electronics and Communication Engineering, SSN College of Engineering, Affiliated to the University of Madras, Chennai, Tamil Nadu, India – 2003

M. Sc., Bioinformatics, University of Madras, Chennai, Tamil Nadu, India – 2005

M. S., Bioinformatics, University of Illinois at Chicago, Chicago, Illinois – 2011

EXPERIENCE: Summer Intern, Dept. of Bioinformatics, Indian Institute of Science, Bangalore, India - May-July 2004

Project Assistant, Dept. of Bioinformatics, National Institute of Immunology, New Delhi, India - Feb-July 2005

Research Assistant, Institute of Bioinformatics and Applied Biotechnology, Bangalore, India - 2006 – 2007

Graduate Student & Research Assistant, Center for Pharmaceutical Biotechnology, UIC, Chicago, IL - 2008 – 2009

Programmer/Analyst I, Department of Genetics & Genomics Sciences, Mount Sinai School of Medicine, New York, NY - 2010 – present

AWARDS: Dr. A.L. Mudaliar Silver Jubilee Endowment Merit Scholar, University of Madras – 2004 – 2005

Best Poster Award, American Society for Biochemistry and Molecular Biology, 2009

