A Dissertation

entitled

The Characterization and Utilization of Middle-range Sequence Patterns within the Human Genome

by

Samuel Steven Shepard

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Doctor of Philosophy Degree in Biomedical Sciences

The University of Toledo
June 2010

An Abstract of

The Characterization and Utilization of Middle-range Sequence Patterns within the Human Genome

by

Samuel Steven Shepard

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Doctor of Philosophy Degree in Biomedical Sciences

The University of Toledo
June 2010

Mid-range inhomogeneity (MRI) is the significant enrichment of particular nucleotides in genomic sequences extending from 30 to 10,000 nucleotides. MRI can be observed for all nucleotide pairings (*e.g.*, G+C, A+G, and G+T) as well as for individual bases. Various types of MRI regions are 4 to 20 times enriched in mammalian genomes compared to their occurrences in random models. We first show how different types of mutations change MRI regions. Human, chimpanzee and *Macaca mulatta* genomes were aligned to study the projected effects of substitutions and indels on human sequence evolution within both MRI regions and control regions of average nucleotide composition. Over 18.8 million fixed point substitutions, 3.9 million SNPs, and indels spanning 6.9 Mb were procured and evaluated in human—1.8 Mb substitutions and 1.9 Mb indels within MRI regions. Ancestral and mutant alleles for substitutions were determined. Substitutions were grouped according to their fixation within human populations: fixed substitutions (from the human-chimp-macaca alignment), major SNPs ($>$ 80% mutant allele frequency within humans), medium SNPs (20%–80%), minor SNPs (3%–20%), and rare SNPs ($<$3%). Data on short ($<$ 3 bp) and medium-length (3–50 bp) insertions and deletions within MRI regions and appropriate control regions were analyzed for their effect on the expansion or diminution of such regions as well as on changing nucleotide composition. MRI regions have comparable levels of *de novo* mutations to the control genomic sequences. Newer mutations rapidly erode MRI regions, bringing their nucleotide composition toward genome-average levels. However, substitutions that favor the maintenance of MRI properties have a higher chance to spread throughout the human population. Indels have a clear tendency to maintain MRI features but have a smaller impact than substitutions. Overall, the observed fixation bias for mutations helps maintain MRI regions during evolution.

Next, we discuss the splicing of large introns in mammals (over 50,000 base-pairs). Large introns must be spliced out of the pre-mRNA in a timely fashion, which involves bringing together distant $5'$ and $3'$ splice sites. In Drosophila large introns can be spliced efficiently through a process known as recursive splicing. We computationally demonstrate that vertebrates lack the proper enrichment of RP-sites in their large introns, and, therefore, require some other method to aid splicing. Over 15,000 non-redundant, large introns from six mammals, 1,600 from chicken and zebrafish, and 560 large introns from five invertebrates were analyzed. Unlike the studied invertebrates, the studied vertebrate genomes contain consistently abundant amounts of direct and complementary strand interspersed repetitive elements (mainly SINEs and LINEs) that may form stems with each other within large introns. Indeed, predicted stems were abundant and stable in the large introns of mammals. We hypothesize that stable stems with long loops within large introns allow splice sites to find each other more quickly by folding the intronic RNA upon itself.

Finally, we extend and complement existing Markov model algorithms by developing and testing a novel binary-abstracted Markov model (BAMM) algorithm. BAMM can emphasize selected portions of genomic sequence signals according to specific abstraction rules. We present abstraction rules that generalize genomic sequence patterns at the single nucleotide level up to the level of tetranucleotides, using both in-frame data and data of mixed reading frames. We develop context-dependent abstraction rules that emphasize genomic sequence repetition. Unlike traditional Markov models, BAMM can analyze nucleotide patterns on the short-range ($<$ 20 bp) up to the mid-range (20 to 50 bp) scale. Abstraction rules can also be both frame sensitive or independent. We build classifiers for both coding sequences and introns as well as for $5'$ and $3'$ UTR data. Using support vector machines, we demonstrate that we can combine multiple BAMM classifiers to get even better exon-intron classification accuracy.

I dedicate this work to my mother, Christy Shepard,

who taught me how to read, how to write,

and who instilled in me the love of learning

and of excellence.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0643542.

I am very grateful to my advisor, Alexei Fedorov, who has been both a great teacher and mentor to me—above and beyond what is called for to do experiments. Furthermore, I am indebted for the lessons I have learned from my other teachers: Mark Borodovsky, Craig Zirbel, and Gursel Serpen, without which graduation would have been far off. The experience and advice from my committee members (Robert Blumenthal, John Gray, Sadik Khuder, and Robert Trumbly) has been most valuable and is deeply appreciated.

Many thanks to Peter Bazeley & Jason Bechtel for their insight into unix, bioinformatics, computer science, and general geekiness. Moreover, without my lab-mates and co-workers—Andrew McSweeny, Dave Rearick, Maryam Nabiyouni, Mark McCreary, Jie He, and especially Ashwin Prakash—I would not have gone far.

I am most thankful to my heavenly Father for His great providence & care, to Christ Jesus for His grace & life, and to the Holy Spirit for His strength, love, and friendship throughout all of life's difficulties. I acknowledge my parents Steven and Christy Shepard, who have always been very supportive of me and who have helped me to both love and pursue truth. I also remember my late grandfather Reynolds S. Shepard (November 27, 1920–December 11, 2009), for the gift of our first computer that started me on my way, for his perpetual interest in my education, and for being a good example of ingenuity and constant curiosity.

Contents

List of Tables

List of Figures

List of Abbreviations

MRI	Mid-range inhomogeneity
BAMM	Binary-abstracted markov model
BPSO	Binary particle swarm optimization
CDS	Coding sequence(s)
indels	Insertions and deletions
MM	Markov model
M-value	Measure of goodness for model accuracy, see Section 4.2.3.1.
mb/Mb, kb, bp	Mega-base, kilobase, and base pair(s) respectively.
nt	Nucleotide
SNP	Single nucleotide polymorphism
SS	Secondary structures
UTR	Untranslated region(s)

Chapter 1

The Consequence of Mid-range Inhomogeneity

1.1 What is Middle-range Inhomogeneity?

It is well-known that DNA sequence patterns within the human genome are not random. Simple dinucleotides such as ApA, CpG, GpG, *et cetera*, appear in the genome at frequencies one would not expect by pure chance. A uniform distribution of 2-mers would predict a frequency of $\frac{1}{16}$; however, this is far from the case in actual genomes, which are nonrandom or *inhomogeneous*. Reasons for this short range nonrandomness are clear: short sequences or words (< 20 nts) have biological meaning, such as in the context of promoter boxes, codons, and the like. Moreover, short range is not the only such inhomogeneity with meaning. Nucleotide sequences many thousands of bases long also have a G+C composition that is nonrandom. Such regions are known as isochores and some of them, depending on their composition, have observed correlations with biological processes such as recombination rate [1].

In 2008 I coauthored a study showing that there also exists a mid-range inhomogneity (about 30 to 1000 basepairs) that is associated with predicted strong, local

RNA secondary structures [2]. Mid-range inhomogeneity or MRI occurs in many different genomic elements including exons, introns, $5'$ and $3'$ UTRs as well as intergenic regions. As a little studied genomic signal, it is important to classify its association with biological meaning, including how it may have been shaped by evolution. The more one elucidates the characteristics of the nonrandom signals in the genome, the better one can apply such knowledge to gene finding and the prediction of other genomic elements.

The euchromatic human genome is 2.85 billlion base-pairs long [3], although if one adds up the lengths of all protein-coding mRNA, regulatory elements, and non-coding RNA with known functions, only about 5% of mammalian genomes are well understood for their usefulness [4]. However, the remaining majority of noncoding DNA is not arranged haphazardly. At the short range level of DNA composition, genomes of different species have particular biases in the frequency distribution of their dinucleotides; that is, to have a short range "genomic signature" ubiquitously throughout their genomes [5]. For example, just within vertebrates, CpC/GpG dinucletides are over-represented in human and cow whereas in rat and mouse TpG/CpA and ApG/CpT dinucleotides are significantly over-represented instead [6]. Other short range nonrandomness exists within the genome besides genomic signatures.

Short range patterns called "pyknons" have been observed throughout the human genome at variable lengths (usually around 17 base pairs) which have special spacing properties within $3'$ UTRs suggestive of biological functions [7, 8]. The genetic code itself also shows short range compositional nonrandomness in that it has a 3 base-pair periodicity with a bias toward the form RNY (R = puRine, N = aNy base, Y = pYrimidine) [9]. In general, we term the nonrandom frequency biases of short (typically < 30 basepairs) oligonucleotides "short range inhomogeneity" (SRI). As one might expect, SRI is not the only kind of inhomogeneity in the human genome.

As noted above, "long-range inhomogeneity" or LRI also exists in the genome in the form of *isochores*. Isochores are contiguous stretches of DNA (usually much greater than 300 kb in length) that maintain a common GC composition even down to a 3 kb window size [10]. According to Bernardi, human isochores can be divided into GC-poor L1 and L2 isochore families and into GC-rich H1, H2, and H3 families. The former group makes up most of the genome and has a low relative abundance of genes within it whereas the latter group makes up a smaller portion of the genome and has a high relative abundance of gene density relative to the size of the regions [11]. Isochores have defined boundaries [12] and have biologically important features according to their family, such as recombination rate [1], in addition to their G+C relationship with gene density. Other statistical studies support the view of the genome as a mosaic of compositionally similar regions.

A statistical tool called *detrended fluctuation analysis* [13] to measure the scale-invariance of the genomic sequence composition, and, thus, the homogeneity of the genome, [14] reveals that human genomic patchiness occurs on the isochore scale as well as on a smaller scale corresponding to genomic elements. This pattern may also differ from species to species [15]. Such statistical data supports the evidence for isochores and is suggestive of a middle-range nonrandomness in the genome.

We define "middle-range inhomogeneity" or MRI as the nonrandom enrichment of nucleotide compositions (A, C, G+C, A+G, etc.) in regions of varying lengths between SRI and LRI (approximately 30 to 1000 nucleotides). For an example, consider any sequence of DNA longer than N nucleotides. Next for some window region of length L, say 50 nucleotides, we might choose a percentage and nucleotide content to test for compositional enrichment (the window can also be extended until it is no longer rich). Suppose we choose the upper threshold to be 80% for G+C nucleotide content, then any non-overlapping window of length 50 bp with at least 40 nucleotides that are G or C will be counted as an MRI content-rich region. Another way to look at MRI regions is by considering the complementary content composition (mathematical complement)–thus, a G+C MRI region is at once both GC-rich or AT-poor, depending on what content type one is most interested in studying.

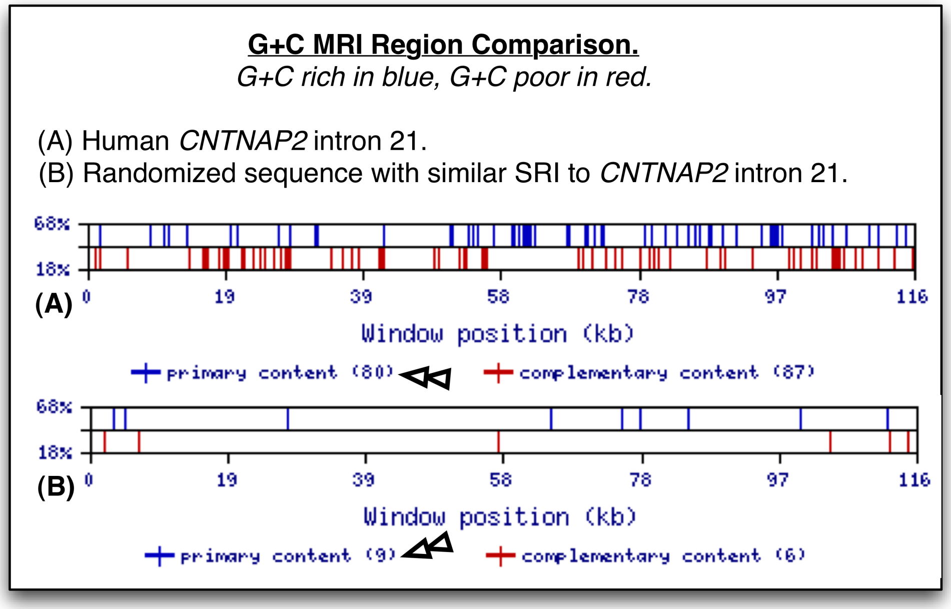


Figure 1-1: A comparison of the MRI regions within human intron 21 for CNTNAP2 **(A)** versus a randomized version of that intron **(B)**. The blue bars represent GC-rich MRI regions while the red bars are for GC-poor MRI regions.

In my first paper on MRI [2] the first task was to demonstrate that MRI regions are not simply a direct by-product of SRI. To show this, my coauthors and I first generated random sequences that approximated the short range inhomogeneity of a natural, genomic sequence (SRI can be quantified by an oligonucleotide frequency table of each fixed length oligonucleotide in a sequence). Second, we showed that the number of MRI regions in the random sequence is considerably less than the number of MRI regions in the natural sequence (from which the random sequence gets its oligonucleotide frequencies, in our case up to the 4-mer level for short sequences). This test was done for various nucleotide content types, including A+G, G+C, and G+T. Figure 1-1 shows an example of the contrast in MRI regions between natural and randomized sequences.

It was found that MRI is a genome-wide phenomenon. Indeed, we have observed MRI in 5′ and 3′-UTRs, introns, exons, and intergenic regions. For each of these regions we were able to show that MRI is associated with predicted strong, local RNA secondary structures. Local RNA secondary structures have interacting nucleotides less than 50 bp apart. Global secondary structures (spanning much longer distances up to hundreds of nucleotides) are more difficult to predict with much reliability [16].

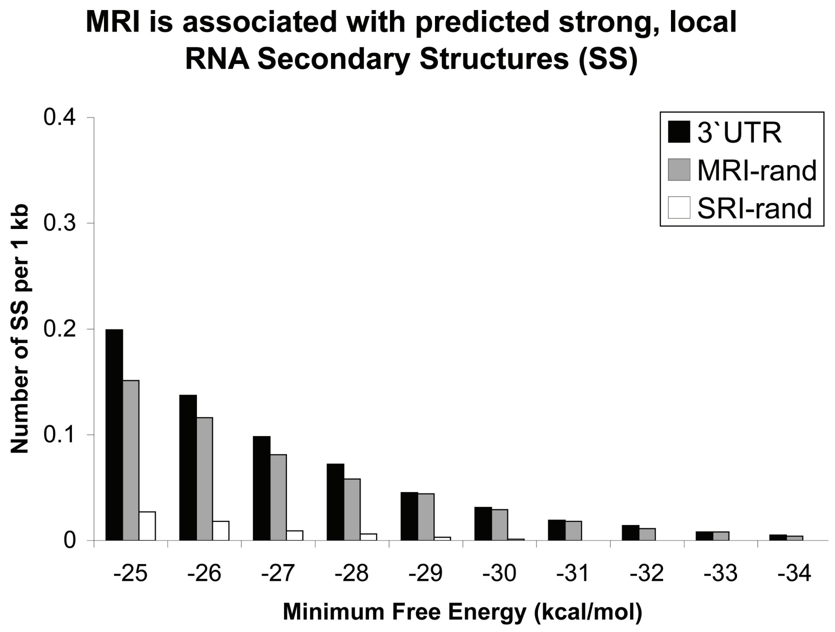


Figure 1-2: The distribution of predicted strong, local RNA secondary structure frequencies in genomic, random MRI-preserving, and random SRI-preserving sequences.

We also created a program to generate randomized sequences that preserved the number and location of MRI regions from an input sequence. Using a genomic sequence, a randomized sequence approximating the SRI of that sequence ("SRI-rand"), and a second randomized sequence that approximates both the SRI and preserves the MRI regions ("MRI-rand") of that sequence, we can compare the number of predicted strong, local RNA secondary structures in each sequence. Observe from Figure 1-2 that the genomic as well as the randomized sequence mimicking MRI are close to each other in terms of the number of strong, local secondary structures while the randomized sequences only approximating the SRI has much fewer structures. Therefore, one possible biological source of MRI is its association with strong local RNA secondary structures. We created an online public resource for studying middle-range inhomogeneity called the Genomic MRI website. It allows users to freely download our programs for private use as well as run their own sequence within the webpage. The reader is invited to visit GMRI at `http://mco321125.meduohio.edu/~jbechtel/gmri/`.

1.2 Studies of the Human Genome

In Chapter 2 my coauthors and I explore the origins of middle-range of inhomogeneity by analyzing different types of mutation within MRI and control regions (regions with an average base composition or no special enrichment). We created a human, chimp, and rhesus monkey genomic triple alignment—studying the projected effects of substitution and indels on human MRI region evolution. In addition to the 18.8 million fixed point substitutions gathered, we analyzed 3.9 million SNPs and 1.9 Mb of indels within MRI and control regions. Original and mutant alleles were determined using the triple alignment for fixed substitutions and a whole genome human-chimp alignment for human SNPs. We grouped substitutions according to the level of fixation within the human population: fixed substitutions (divergence of human from rhesus and chimp), major SNPs ($> 80\%$ mutant allele frequency in humans), medium SNPs (20%–80%), minor SNPs (3%–20%), and rare SNPs ($< 3\%$ mutant allele within humans). Short insertion-deletions (< 3 bp) and medium-length indels (3–50 bp) were also studied for their contribution to the enrichment or erosion of MRI regions within humans. We demonstrated that the levels of *de novo* mutations were similar in both the MRI regions and the control genomics sequences. The projected trend of such new mutations tends to erode MRI regions and bring the nucleotide composition of MRI regions toward a more genome-average level. We found, however, that mutations that were increasingly fixed within the human population tended to maintain and not erode the enrichment of MRI regions. Insertions and deletions also tend to maintain MRI regions although they are not as potent in their effect as substitutions. Since the fixation bias of these mutations within MRI regions tends to preserve the compositional enrichment rather than erode it, we believe MRI is under weak selection due to its association with a variety of biologically important elements such as RNA secondary structures.

For Chapter 3 I study a special class of predicted RNA secondary structures called "hairpins" that may affect the splicing of "large introns" (> 50 kb). Large introns must be spliced out of the pre-mRNA in a timely fashion, but this must be difficult given the distance of $5'$ and $3'$ acceptor and donor splice sites. In Drosophila, this problem is solved by a process known as "recursive splicing"—a consecutive splicing from the $5'$-end at a series of combined donor-acceptor splice sites called RP-sites [17]. We show that vertebrates lack the proper enrichment of RP-sites within their large introns, and so must require some *other* method to aid their large intron splicing. We inspected over 15,000 non-redundant, large introns from six mammals, 1,600 from chicken and zebrafish, and 560 non-redundant large introns from five invertebrates. The investigation demonstrates that the studied vertebrate genomes contain consistently abundant amounts of direct and complementary strand interspersed repetitive elements (mainly SINEs and LINEs) that have potential to form stems with each other within large introns. The predicted stems are shown to be abundant and stable within the large introns of mammals. Perhaps stems with long loops, as would be needed in the context of large introns, would allow the intron splice sites to find each other more quickly by folding the intronic RNA upon itself at intervals. Since *Alu* repeats are enriched by C/G bases in certain portions of their sequence [2], the aid of hairpin structures for large intron splicing may be thought of as another association and consequence of middle-range inhomogeneity.

Given the biological significance of middle-range sequence patterns, especially within the context of introns, Chapter 4 extends the traditional homogeneous nucleotide Markov model algorithm to be able to analyze human genomic sequences on the middle-range scale using a “fuzzy sequence” or nucleotide to binary abstraction process. Our “binary-abstracted Markov model” (BAMM) algorithm has an effective nucleotide coverage both on the mid-range scale (20 to 50 bp) as well as on the short range ($<$ 20 bp) scale and is able to be used to discriminate between coding exon and introns. Moreover, because of the information reduction achieved by the nucleotide to binary abstraction process, binary-abstracted Markov models are able to emphasize different biologically-relevant genomic sequence patterns using different nucleotide “abstraction rules” on the *same* genomic sequence. We show abstraction rules that are frame-independent (when considering coding sequences), ones that depend on the reading frame, ones that emphasize patterns in sequence repetition, and ones that emphasize splicing signals. We also demonstrate that BAMM can be used not only to classify coding exons and introns, but also to discriminate $5'$ untranslated exons and introns. We believe that our method will help complement existing gene prediction methodologies as well as aid in the study of important middle-range genomic sequence patterns.

References

[1] J. I. Montoya-Burgos, P. Boursot, and N. Galtier, “Recombination explains isochores in mammalian genomes.,” *Trends Genet*, vol. 19, no. 3, pp. 128–130, 2003 Mar.

[2] J. M. Bechtel, T. Wittenschlaeger, T. Dwyer, J. Song, S. Arunachalam, S. K. Ramakrishnan, S. Shepard, and A. Fedorov, “Genomic mid-range inhomogeneity correlates with an abundance of rna secondary structures.,” *BMC Genomics*, vol. 9, p. 284, 2008.

[3] I. H. G. Consortium, “Finishing the euchromatic sequence of the human genome.,” *Nature*, vol. 431, no. 7011, pp. 931–945, 2004 Oct 21.

[4] L. Fedorova and A. Fedorov, “Puzzles of the human genome: Why do we need our introns?,” *Current Genomics*, vol. 6, pp. 589–595, DEC 2005.

[5] S. Karlin and C. Burge, “Dinucleotide relative abundance extremes: a genomic signature.,” *Trends Genet*, vol. 11, no. 7, pp. 283–290, 1995 Jul.

[6] S. Karlin and J. Mrazek, “Compositional differences within and between eukaryotic genomes.,” *Proc Natl Acad Sci U S A*, vol. 94, no. 19, pp. 10227–10232, 1997 Sep 16.

[7] I. Rigoutsos, T. Huynh, K. Miranda, A. Tsirigos, A. McHardy, and D. Platt, “Short blocks from the noncoding parts of the human genome have instances within nearly all known genes and relate to biological processes.,” *Proc Natl Acad Sci U S A*, vol. 103, no. 17, pp. 6605–6610, 2006 Apr 25.

[8] A. Meynert and E. Birney, “Picking pyknons out of the human genome.,” *Cell*, vol. 125, no. 5, pp. 836–838, 2006 Jun 2.

[9] J. C. Shepherd, “Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification.,” *Proc Natl Acad Sci U S A*, vol. 78, no. 3, pp. 1596–1600, 1981 Mar.

[10] G. Bernardi, “Isochores and the evolutionary genomics of vertebrates.,” *Gene*, vol. 241, no. 1, pp. 3–17, 2000 Jan 4.

[11] G. Bernardi, “The vertebrate genome: isochores and evolution.,” *Mol Biol Evol*, vol. 10, no. 1, pp. 186–204, 1993 Jan.

[12] T. Fukagawa, K. Sugaya, K. Matsumoto, K. Okumura, A. Ando, H. Inoko, and T. Ikemura, “A boundary of long-range g + c% mosaic domains in the human mhc locus: pseudoautosomal boundary-like sequence exists near the boundary.,” *Genomics*, vol. 25, no. 1, pp. 184–191, 1995 Jan 1.

[13] C. K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger, “Mosaic organization of dna nucleotides.,” *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, vol. 49, no. 2, pp. 1685–1689, 1994 Feb.

[14] P. Carpena, P. Bernaola-Galvan, A. V. Coronado, M. Hackenberg, and J. L. Oliver, “Identifying characteristic scales in the human genome.,” *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 75, no. 3 Pt 1, p. 032903, 2007 Mar.

[15] J. L. Oliver, P. Bernaola-Galvan, M. Hackenberg, and P. Carpena, “Phylogenetic distribution of large-scale genome patchiness.,” *BMC Evol Biol*, vol. 8, p. 107, 2008.

[16] D. H. Mathews, “Predicting a set of minimal free energy rna secondary structures common to two sequences.,” *Bioinformatics*, vol. 21, no. 10, pp. 2246–2253, 2005 May 15.

[17] J. M. Burnette, E. Miyamoto-Sato, M. A. Schaub, J. Conklin, and A. J. Lopez, "Subdivision of large introns in drosophila by recursive splicing at nonexonic elements," *Genetics*, vol. 170, pp. 661–74, Jun 2005.

Chapter 2

Evolution of genomic sequence inhomogeneity at mid-range scales

Authors:

Ashwin Prakash^{†,1,7}, Samuel S. Shepard^{†,1,7},

Jie He[2],Benjamin Hart[3], Miao Chen[3],

Surya P. Amarachintha[4], Olga Mileyeva-Biebesheimer[5],

Jason Bechtel[6], and Alexei Fedorov$^{+,6,7}$

†Equal author contribution.
+Corresponding author.

[1] Program in Cardiovascular & Metabolic Diseases Track, Biomedical Sciences, University of Toledo, Toledo, OH 43614, USA

[2] University of Toledo, Department of Biology, Toledo, Ohio

[3] Dept of Medical Microbiology & Immunology, Infection, Immunity & Transplantation Track, University of Toledo, Toledo, OH 43614.

[4] Department of Biological Sciences, Bowling Green State University, Bowling Green, OH - 43403.

[5] Department of Civil Engineering, University of Toledo, Toledo, Ohio.

[6] Program in Bioinformatics and Proteomics/Genomics, University of Toledo, Toledo, OH 43614, USA.

[7] Department of Medicine, University of Toledo, Toledo, Ohio.

(Published in *BMC Genomics.* 2009 Nov 5;10:513.)

Abstract

Background: Mid-range inhomogeneity or MRI is the significant enrichment of particular nucleotides in genomic sequences extending from 30 up to several thousands of nucleotides. The best-known manifestation of MRI is CpG islands representing CG-rich regions. Recently it was demonstrated that MRI could be observed not only for G+C content but also for all other nucleotide pairings (e.g. A+G and G+T) as well as for individual bases. Various types of MRI regions are 4-20 times enriched in mammalian genomes compared to their occurrences in random models.

Results: This paper explores how different types of mutations change MRI regions. Human, chimpanzee and *Macaca mulatta* genomes were aligned to study the projected effects of substitutions and indels on human sequence evolution within both MRI regions and control regions of average nucleotide composition. Over 18.8 million fixed point substitutions, 3.9 million SNPs, and indels spanning 6.9 Mb were procured and evaluated in human. They include 1.8 Mb substitutions and 1.9 Mb indels within MRI regions. Ancestral and mutant (derived) alleles for substitutions have been determined. Substitutions were grouped according to their fixation within human populations: fixed substitutions (from the human-chimp-macaca alignment), major SNPs ($>$ 80% mutant allele frequency within humans), medium SNPs (20% - 80% mutant allele frequency), minor SNPs (3% - 20%), and rare SNPs ($<$3%). Data on short ($<$ 3 bp) and medium-length (3 - 50 bp) insertions and deletions within MRI regions and appropriate control regions were analyzed for the effect of indels on the expansion or diminution of such regions as well as on changing nucleotide composition.

Conclusions: MRI regions have comparable levels of de novo mutations to the control genomic sequences with average base composition. De novo substitutions rapidly erode MRI regions, bringing their nucleotide composition toward genome-

average levels. However, those substitutions that favor the maintenance of MRI properties have a higher chance to spread through the entire population. Indels have a clear tendency to maintain MRI features yet they have a smaller impact than substitutions. All in all, the observed fixation bias for mutations helps to preserve MRI regions during evolution.

2.1 Introduction

The protein coding sequences of humans and of most other mammals represent less than 2% of their genomes. The remaining 98% is made up of 5′- and 3′-untranslated regions of mRNAs (<2%), introns (∼37%), and intergenic regions (∼60%) [1]. These vast non-protein coding genomic areas, previously frequently referred to as "junk" DNA, contain numerous functional signals of various origin and purpose. They include thousands of non-protein coding RNAs [2], numerous gene expression regulatory signals that surround each gene, chromatin folding structures which include nucleosome positioning sites and scaffold/matrix attached regions [3, 4]. These functional DNA regions are non-random in their genomic sequence. The non-randomness or inhomogeneity of base composition has been described at different levels of complexity and sequence length. Starting on the short scale, inhomogeneity occurs in the non-random associations of neighboring bases with each other [5], through the over and under-abundance of particular "words" (usually 5-10 base long oligonucleotides) [6] or longer stretches of DNA, also known as "pyknons" (∼ 18 bases long) [7, 8], and up to large regions that cover hundreds of thousands of nucleotides [9]. Compositional inhomogeneity is known to exist in all kinds of species from bacteria to human. However, the particular arrangement of such sequence patterns is often species-specific [10].

It has been the focus of our research to elucidate the genomic sequence non-randomness that we call Mid-Range Inhomogeneity or MRI [11]. We define MRI to be genomic regions from 30 bp to several thousand nucleotides with particular nucleotide enrichments. For large mammalian genomes, there is a high probability that a random sequence of length 20 nucleotides will be unique. Thus, for examining mid-range genomic signals we do not look at particular "words" but only the overall compositional content of particular base(s) that we refer to as X (X could be a single nucleotide A, G, C, or T or any of their combinations like A+C, or G+T+C). We created a public Internet resource, "Genomic MRI" to study the distribution of X-rich regions in any sequence of interest. It was demonstrated that X-rich MRI regions are highly overrepresented in mammalian genomes for all kinds X-contexts. Particular properties of MRI have also been investigated previously by Mrazek and Kypr [12] and also by Nikolaou and Almirantis [13]. This paper studies the effect of mutations on the evolution of MRI regions in primates.

2.2 Results

2.2.1 Substitution and polymorphism inside MRI regions

From the whole-genome human-chimp-macaque alignment we extracted all the aligned sequences with inhomogeneous nucleotide compositions that satisfy the criteria for MRI (so-called X-rich MRI regions; see the Materials and Methods section) and also control regions with nucleotide compositions equal to the average values for the entire human genome. We used the default MRI region length of 100 nucleotides for all computations. Only SNPs located within these MRI and control regions were studied. We particularly focused on the single nucleotide substitutions that maintain or erode MRI features. For example, in GT-rich MRI regions we counted the total number of novel polymorphisms that erode the feature, i.e. G or T $\rightarrow$ C or A substitutions, denoted as $N_{GT\rightarrow CA}$ and also the total number of those that maintain the MRI features, i.e. C or A $\rightarrow$ G or T substitutions, denoted as $N_{CA\rightarrow GT}$. In addition, the entire set of recent human substitutions; that is, those nucleotides that differed in human but were the same in chimp and macaque, were processed for the MRI and control regions and presented as "fixed substitutions". The substitution ratio, S_X (recall that: $S_X = N_{X \to nonX}/N_{nonX \to X}$) for the numbers of substitutions that maintain and erode X-rich MRI features was calculated for each substitution subtype (rare, minor, medium, and major SNPs and 'fixed'—refer to the Methods section for a detailed explanation) and presented in Figures 2-1 and 2-2. With respect to X-rich or poor MRI regions, the X in Figure 2-1 represents a two base combination such as GC, AG, GT, etc. while in Figure 2 X can be any single nucleotide, e.g., A,T,C, and G. If the S_X-ratio is equal to 1 the X-rich region does not tend towards a change in its X-base composition. When $S_X > 1$, the substitutions reduce the X-richness of the examined regions, whereas when $S_X < 1$, substitution rates elevate the X-richness of the regions. Figures 2-1 and 2-2 demonstrate clear linear trends for S_X-ratios with respect to increasing fixation of substitutions within human populations.

For the cases of GT-, AC-, AG-, and TC-rich MRI regions (Figure 2-1), all S-ratios for rare SNPs are close to 1.8 (showing erosion of the MRI features). For major SNPs and fixed mutations the S_{GT} and S_{AC}-ratios reach 1.0 (which means no change in the corresponding base composition) and S_{GA} and S_{CT}-ratios reach 1.2 respectively. As for the corresponding control GT-, AC-, AG-, and TC-average regions (all having 50% of corresponding base composition) these lines are flat with all S-ratios equal to 1. The latter result is highly expected because of the symmetry of (+) and (−) chromosomal strands for these particular base compositions. Figure 1 also demonstrates that in GC-rich MRI regions the S_{GC}-ratio change has the highest slope from 7.0 for rare SNPs to 1.6 for fixed substitutions. In AT-rich MRI regions (also referred to as nonGC-rich in the tables) the change of S_{AT}-ratio has the lowest slope starting from 1.7 (rare SNPs) and ending at 1.3 (fixed substitutions). The control regions with the average GC/AT compositions (40-42% GC and 58-60% AT) also demonstrate a clear

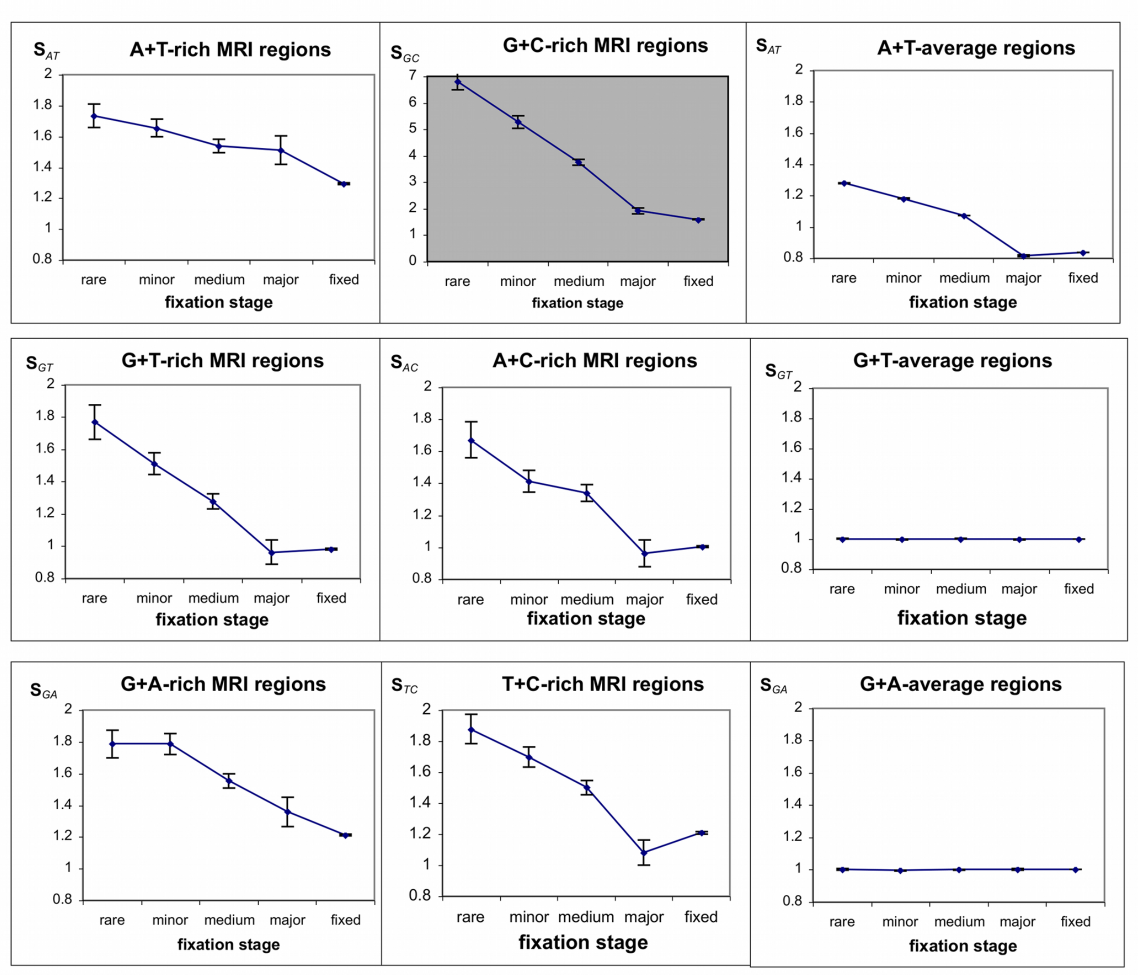


Figure 2-1: For each X MRI region—where X is for GC-, GT-, or GA-rich or poor regions—the X-base composition rate of change is given for all substitutions at different levels of fixation within the human population. The rate of change (S_X) is the ratio of X to $nonX$ substitutions over $nonX$ to X substitutions in those particular X-rich regions. Thus, a ratio of 1 means no change in the X-richness of the region whereas a ratio greater than 1 implies degradation of the X-rich region and less than 1 implies enrichment of the X-rich MRI region. Note that in the control X-average regions the S_X-ratio is always inverse to S_{nonX}-ratio ($S_X = 1/S_{nonX}$). Therefore, only one graph for each S_X and S_{nonX} pair is presented. Since there are significant variations in S_X-ratios for different X compositions, the graphs are presented in two different scales. The white background presents changes of S_X-ratios in the 0.8 to 2 range, while the gray background presents changes in the 0 to 7 range. Vertical bars show the standard error of the means (see Methods section).

change of S-ratios during substitution fixation. In the control GC-average regions, rare SNPs favor increasing AT-richness (S_{GC}-ratio of 1.3) whereas fixed mutations demonstrate the opposite effect (S_{GC}-ratio of 0.8).

The data for the S-ratios for single nucleotides (Figure 2-2) are very similar to the trends seen in GC- and AT-rich regions. As expected from (+/-) strand symmetry, S_G-ratios are equal to S_C-ratios and represent about a half of the GC trend. The minor differences between G- and C-rich regions are within the errors of measurement. In the same way the S_A-ratios are seen to be the same as the S_T-ratios and they comprise approximately half of the effect seen for AT-rich regions.

Based on the observed S_X-ratios and the current percentage of X bases in the genomic regions under investigation, we calculated the projected equilibrium composition representing the future X-composition toward which the examined substitution rates drive these regions. In other words, the equilibrium X-composition shows the future level of X-richness that would be approached if the S_X-ratio as it is observed now were maintained indefinitely. The computed equilibria for each subgroup of substitutions are presented in Table 2.1. For instance, in the GA-rich regions (G+A composition of 70%), rare SNPs drive GA-richness of these MRI regions down to an equilibrium of 56.6%, while nearly fixed or fixed substitutions drive the GA-composition only to the 65.8% level. For each type of X-rich MRI, there is a trend toward minimizing the damage of mutations and preserving the MRI feature as the fixation of the observed substitutions increases. The highest preservation effect is seen for GT- and AC-rich regions (with an observed X-base composition of 70%), where the equilibria for fixed substitutions reach about the same level of 70%. For the rest of the types of MRI regions, their equilibria composition is a little below the currently observed base composition.

In order to estimate mutation rates for MRI regions versus their respective control regions, we counted the occurrence rates for rare SNPs. The frequency ratio of rare

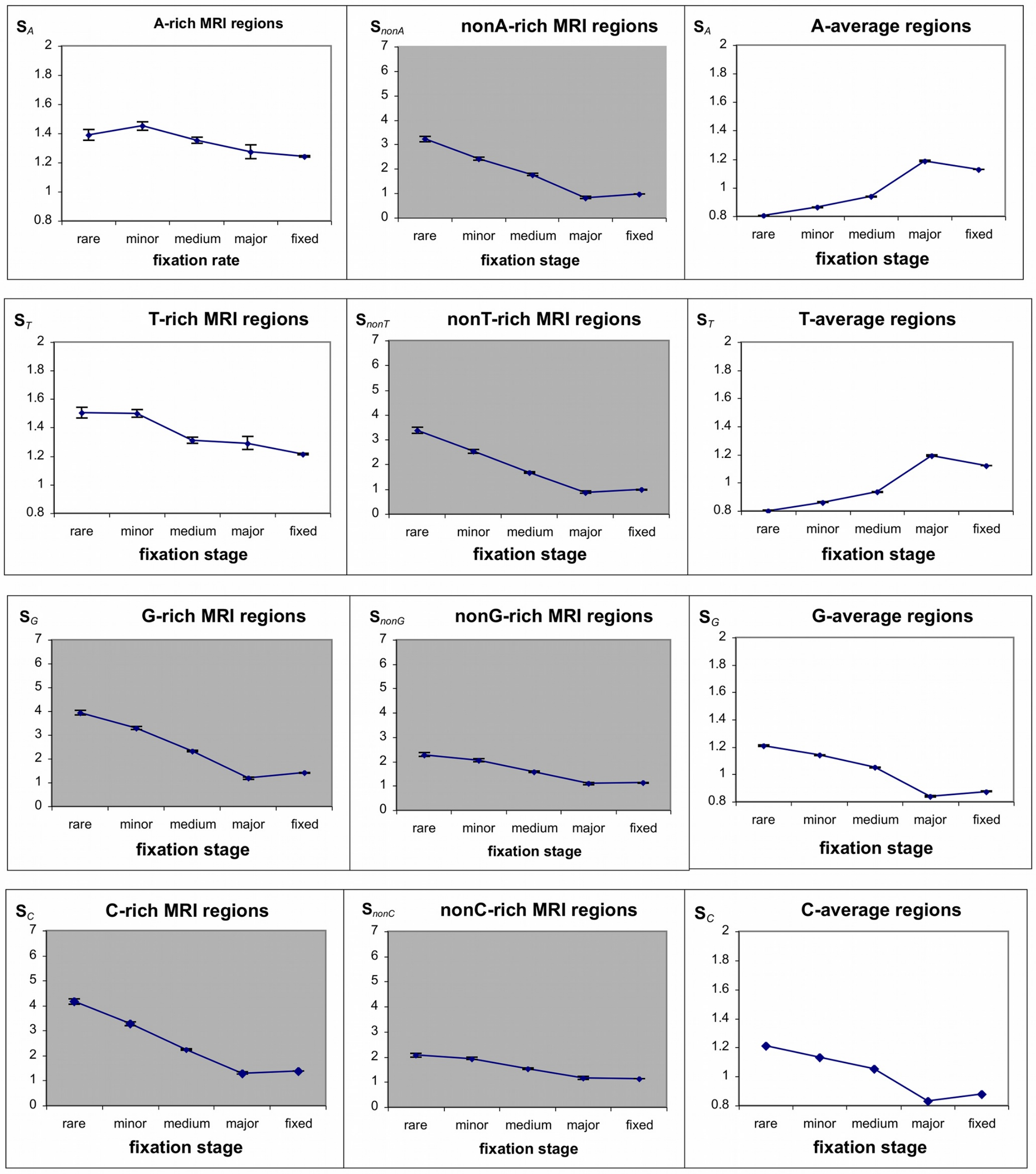


Figure 2-2: For each X MRI region—where X is for A-, T-, G-, or C-rich or poor regions—the X-base composition rate of change is given for all substitutions at different levels of fixation within the human population. The rate of change (S_X) is the ratio of X to $nonX$ substitutions over $nonX$ to X substitutions in those particular X-rich regions. Thus, a ratio of 1 means no change in the X-richness of the region whereas a ratio greater than 1 implies degradation of the X-rich region and less than 1 implies enrichment of the X-rich MRI region. Note that in the control X-average regions the S_X-ratio is always inverse to S_{nonX}-ratio ($S_X = 1/S_{nonX}$). Therefore, only one graph for each S_X and S_{nonX} pair is presented. Since there are significant variations in S_X-ratios for different X compositions, the graphs are presented in two different scales. The white background presents changes of S_X-ratios in the 0.8 to 2 range, while the gray background presents changes in the 0 to 7 range. Vertical bars show the standard error of the means (see Methods section).

	Equilibrium for X-percentage computed from each substitution rate					
Type of region	Observed X-percentage	rare SNPs	minor SNPs	Medium SNPs	major SNPs	fixed substit.
G-rich	40%	14.5%	16.9%	22.3%	36.0%	32.1%
nonG-rich	7%	14.6	13.4	10.6	7.6	7.8
G-average	20%	17.1	18.0	19.2	23.0	22.2
C-rich	40%	13.8	16.9	23.0	33.90	32.4
nonC-rich	7%	14.5	12.7	10.2	8.0	7.8
C-average	20%	17.1	18.1	19.2	23.1	22.1
A-rich	49.5%	41.4	40.3	42.0	43.5	44.1
nonA-rich	12.9%	32.3	26.3	20.8	10.9	12.6
A-average	29.4%	34.1	32.6	30.7	26.0	27.0
T-rich	49.5%	39.5	39.5	42.8	43.2	44.6
nonT-rich	12.9%	33.4	27.3	19.9	11.5	12.6
T-average	29.4%	34.2	32.6	30.8	25.9	27.1
GT-rich	69.8%	56.9	60.7	64.6	70.8	70.4
nonGT-rich	30.1%	41.7	37.7	36.5	29.2	30.1
GT-average	50.0%	49.9	50.0	50.0	50.0	50.0
GA-rich	70.0%	56.6	56.6	60.0	63.2	65.8
nonGA-rich	29.9%	44.6	42.1	39.1	31.7	34.1
GA-average	50.0%	49.9	50.1	50.0	49.9	49.9
GC-rich	71.3%	26.4	31.7	39.5	56.1	60.6
nonGC-rich	20.0%	30.2	29.3	27.8	27.4	24.4
GC-average	40.7%	34.9	36.8	39.0	45.7	45.0

Table 2.1: The calculated equilibria percentages (see Equation 2.3) for X-bases in X-rich MRI and control regions with average X-composition. Projected equilibria are given based on the substitution rates of rare, minor, medium, and major SNPs as well as for the fixed substitution rates (chimp-macaque to human).

SNPs in MRI rich regions to those in the control regions was calculated. The smallest ratio observed was for A+C content (0.464). This means the frequency of rare SNPs within MRI AC-rich regions is approximately half that of control regions. The highest occurrence ratio for rare SNPs was observed in G- and C-rich MRI regions (1.16 and 1.17 respecitvely). Thus, the occurrence rates of rare SNPs is slightly lower in MRI regions than in the corresponding control regions with the exception of G- and C-rich MRI regions. The entire dataset for the SNPs occurrences in MRI and control regions is presented in Supplementary Table 2.4. The prevalence of rare and minor SNPs over major SNPs was also observed, their proportion over every MRI and control regions being 5.79.

2.2.2 Insertions and deletions inside MRI regions

Using the same computational approach as for substitutions, we analyzed human-chimp-macaque triple alignments for the characterization of *indels* (insertions & deletions) that occurred in the human genome during the last 10 million years after the divergence of *H. sapiens* and *P. troglodytes* species. We particularly investigated how indels change the nucleotide composition of MRI regions and control regions with average nucleotide composition. The complete set of data representing short indels (whose sizes are less than three nucleotides) and medium indels (whose sizes are from three to fifty nucleotides) is presented in Supplementary Table 2.5. Large indels with sizes over 50 bp were not examined since they are comparable with the sizes of MRI regions and, thus, compromise proper characterization of MRI. The summary data on the influence of both short and medium indels on the composition of MRI and control regions are presented in the Tables 2.2 and 2.3 (Table 2.2 shows MRI regions where X represents any single nucleotide; Table 2.3 is for when X represents any combination of 2 nucleotides). For each type of X-rich and X-control regions the total number of inserted and deleted X and $nonX$ nucleotides have been computed: $N_{ins(X)}$, $N_{ins(nonX)}$, $N_{del(X)}$, $N_{del(nonX)}$. Finally, the net change in X and $nonX$ compositions due to indels have been calculated using the following formulas:

$$\Delta X = N_{ins(X)} - N_{del(X)}$$

$$\Delta nonX = N_{ins(nonX)} - N_{del(nonX)}$$

Tables 2.2 and 2.3 demonstrate that in the human genome there is a prevalence of deletions over insertions (i.e. negative values of ΔX and $\Delta nonX$) for every type of nucleotide content studied and for every type of MRI and control region with the exception of GC-indels in GC-rich MRI regions. In the last case ΔGC is positive and equal to 1405 added nucleotides (over a total set of 1.8 million nucleotides). For all other cases of X except X =GC, short and medium indels cause gradual contraction of genomic regions in humans. This means that there is no nucleotide composition equilibrium to which the indels drive the genome in the indefinite future and, therefore, these equilibria have not been calculated. Table 2.2 shows that, for every X-rich region, indels result in the increasing the richness of corresponding MRI regions (positive net $X\%$ change for X-rich region and negative net $X\%$ change for $nonX$-rich region). In all X-control regions the net $X\%$ change is several times less than in the corresponding X-rich and $nonX$-rich regions.

Finally, we calculated the percentage of nucleotide composition changes in case of both substitutions and indels separately, that occurred in the human genome during last ten million years after the divergence of human and chimpanzee. These results are presented in Tables 2 and 3 and serve to measure the relative importance of substitutions versus indels to the nucleotide composition of MRI regions.

2.3 Discussion

Consistent with Chargaff's second parity rule [14], both the G or C base content of the human genome are equal to 21.1%, while A or T comprise 28.9% each. However, in thousands and thousands of genomic regions of various lengths, the composition of A, T, C, or G content (or different combinations of these bases) exist at extremes quite different from the aforementioned averages. De novo mutations constantly occur in populations and could dramatically change the base composition of a genomic region during the course of evolution. A good choice for a large-scale computational analysis of these novel mutations is in the examination of 'rare' single-nucleotide poly-

	A-rich	nonA-rich	A-average
total length	66.9 Mb	72.4 Mb	800.4 Mb
content of A	49.6%	12.9%	30.5%
ΔA	-16850	-7390	-44182
ΔnonA	-24748	-29257	-98769
net A% change INDEL	0.006%	-0.004%	-0.0001%
net A% change SUBST	-0.027%	-0.002%	-0.014%
	T-rich	nonT-rich	T-average
total length	67.8 Mb	71.1 Mb	800.4 Mb
content of T	49.5%	13.1%	30.5%
ΔT	-21849	-7078	-47238
ΔnonT	-24084	-22716	-97057
net T% change INDEL	0.001%	-0.004%	-0.0004%
net T% change SUBST	-0.024%	-0.002%	-0.013%
	G-rich	nonG-rich	G-average
total length	52.0 Mb	60.4 Mb	884.7 Mb
content of G	40.10%	7.20%	20.40%
ΔG	-1185	-7080	-31780
ΔnonG	-12864	-37512	-139126
net G% change INDEL	0.009%	-0.006%	0.0003%
net G% change SUBST	-0.052%	0.009%	0.016%
	C-rich	nonC-rich	C-average
total length	52.0 Mb	60.4 Mb	883.9 Mb
content of C	40.10%	7.20%	20.50%
ΔC	-829	-6700	-33823
ΔnonC	-12418	-35277	-140331
net C% change INDEL	0.009%	-0.006%	0.0002%
net C% change SUBST	-0.049%	0.009%	0.015%

Table 2.2: The impact of indels on X-rich MRI regions and on X-average regions, where X is for A-, T-, C-, or G-rich or poor. For each particular region we give the total length of examined regions in mega-bases, the percentage composition or content of X, the number of changes in X due to insertions and deletions ($\Delta X = N_{ins}(X) - N_{del}(X)$), and the net change in X composition due to both indels and substitutions.

morphisms (SNPs, or mutations that are present only in a small group of individuals and absent in a majority of the population). Rare SNPs are mutations that have recently occurred. However, even among rare SNPs there exists a minor subgroup of "older" mutations that have diminished their frequency to rare events. The relative size of this subgroup is in reverse proportion to the effective size of the population [15],

	GC-rich	nonGC-rich	GC-average
total length	17.8 Mb	54.8 Mb	780.6 Mb
content of GC	71.00%	20.30%	40.90%
ΔGC	1405	-9100	-31622
ΔnonGC	-765	-5951	-56278
net GC% change INDEL	0.005%	-0.011%	0.001%
net GC% change SUBST	-0.094%	0.042%	0.034%
	GT-rich	nonGT-rich	GT-average
total length	34.9 Mb	34.6 Mb	1192 Mb
content of GT	69.10%	30.90%	50.00%
ΔGT	-8278	-6837	-121644
ΔnonGT	-4518	-8502	-120128
net GT% change INDEL	0.002%	-0.006%	-0.0001%
net GT% change SUBST	0.004%	0.001%	-0.0003%
	GA-rich	nonGA-rich	GA-average
total length	69.2 Mb	70.0 Mb	978.3 Mb
content of GA	69.75%	30.22%	49.99%
ΔGA	-23641	-13935	-96617
ΔnonGA	-14185	-28480	-100013
net GA% change INDEL	0.004%	-0.002%	0.0002%
net GA% change SUBST	-0.014%	0.014%	0.0002%

Table 2.3: The impact of indels on X-rich MRI regions and on X-average regions, where X is for GC-, GT-, or GA-rich or poor. For each particular region we give the total length of examined regions in mega-bases, the percentage composition or content of X, the number of changes in X due to insertions and deletions ($\Delta X = N_{ins}(X) - N_{del}(X)$), and the net change in X composition due to both indels and substitutions.

and hence, it represents only a minor fraction of the recent mutations for humans. Here we show that rare SNPs in genomic regions with average nucleotide composition are enriched by G or C $\rightarrow$ T or A substitutions that drive the genomic composition of those regions to a level of 35% for G+C and 65% for A+T. On the other hand, examining the same regions for mutations that have substantially propagated into human populations (i.e. medium and high frequency SNPs as well as "fixed" recent mutations) demonstrates that these fixed or nearly fixed substitutions are much less prone to G or C $\rightarrow$ T or A changes. Instead, high frequency SNPs as well as fixed substitutions tend to drive genomic regions with average base composition to 45% G+C composition.

Here we have focused particularly on the influence of mutations on the evolution of specific genomic regions with strongly inhomogeneous base compositions that are far from the average distribution of nucleotides (so-called MRI regions where G+C, G+A, C+T, G+T, or A+C composition is at least 70%, A+T composition is above 80%, or single base frequency reaches nearly 50%). For all types of MRI regions, we found that novel substitutions (rare SNPs) tend to more strongly erode the compositional extremes (X-richness) of the region. At the same time, these mutations undergo a strong fixation bias during their propagation into populations in such a way that fixed substitutions tend to preserve MRI regions. For example, rare SNPs inside GC-rich MRI regions drive the nucleotide composition of those regions to the 26% GC level. However, fixed substitutions in the same GC-rich MRI regions drive GC composition only to 61%. The highest fixation was seen for GT- and AC-rich MRI regions, which preserves the current GT- and AC-composition of 70%.

This trend of preserving nucleotide composition of MRI regions with respect to the increasing fixation of substitutions could be explained by at least two different mechanisms. First, one could observe that there are some important functional roles for MRI regions. For instance, GC-rich MRI regions include well-known CG-islands, prominent regulators for gene expression [16,17]. Thus, these regions should be under the constraint of purifying selection, preserving their important features. Other MRI regions may be under similar selective pressure due to association with functional genomic elements and/or, as yet unknown, sequence signals. Second, fixation bias inside MRI regions might be due to some non-symmetry in cellular molecular machinery involving DNA repair, replication, and/or recombination processes. For example, the Biased Gene Conversion (BGC)-theory engages this particular scenario in order to explain the maintenance of CG-rich regions [18,19]. (It must be observed, however, that this theory operates on much larger genomic scales and refers to isochores that cover from hundreds of thousands to millions of bases.) Thus far it is inconclusive as to which of these two scenarios, or a combination thereof, best fits the observed trends. For the case of GC-rich sequences, we conjecture that both scenarios could be taking place to some extent to preserve MRI.

Interestingly, the highest level of MRI erosion for rare SNPs is observed in GC-rich MRI regions. Novel substitutions in these particular regions try to drive GC-content to the lowest level of 26% (see Table 2.1). We explain this phenomenon via uneven distribution of CpG dinucleotides, which are most abundant in GC-rich MRI regions. It is well known that CpG dinucleotides are extreme hot spots for the C$\rightarrow$T and G$\rightarrow$A mutations, which cause CpG to be the most underrepresented dinucleotide in vertebrate genomes. Therefore, CG-rich MRI regions, which are known to have the highest concentration of CpG dinucleotides, should have the highest rate of de novo mutations in the direction C or G $\rightarrow$ T or A. Human SNPs having C/T alleles in the CpG/TpG context with the orthologous chimp allele in the TpG context have an increased error rate of 9.8% for ancestral misidentification (see the Methods section) due to the probability of a coinciding chimp SNP at the same locus [20]. However, since the strength of the mutational erosion in the GC-rich MRI regions is so high, even an error rate of 9.8% will not change the observed trend.

So far we have discussed only the effect of substitutions on the nucleotide composition of mid-range genomic regions. Insertions and deletions are the other types of mutations that change genomic sequences and, therefore, should also be considered. In mammals, short and medium indels are several times less frequent than substitutions. Currently, there is not enough data on human indel SNPs to perform the same analysis of their fixation process as we did for substitutions. For this reason we studied only fixed indels in humans (indels present in human but differing in chimp and macaque). Our examination demonstrated that indels weakly influence the nucleotide content of MRI regions toward preserving their inhomogeneous composition, in the same manner as the fixation bias of fixed substitutions (see Tables 2 and 3).

2.4 Conclusions

The fixation bias on both fixed substitutions and indels tend to protect MRI regions from degradation of their compositional extremes amid the constant flow of random mutations, thus suggesting their contribution in the preservation of functional and structural complexities of the human genome. Future research on these genomic elements as well as refinement of our approach should help determine the extent of maintenance of MRI by natural selection.

2.5 Methods

2.5.1 Genomic samples and computation of recent human mutations ("fixed substitutions").

Taking human-chimp (human build 36.1 and chimp build 2 version 1) and human-macaque (macaque build v1 edit4) whole-genome pairwise alignments from the UCSC Genome Browser [21] (`http://hgdownload.cse.ucsc.edu/downloads.html`) as input, we generated a Perl script for the identification of the common genomic regions for these three species. The process involved the usage of the human genomic sequence as the reference for the location with the chimp and macaque sequences being extracted only in areas where the sequences of all three species were represented. We then invoked the ClustalW (v1.83) program with default parameters to obtain a whole-genome human-chimpanzee-macaque triple alignment. The obtained alignment is available at our website (`http://bpg.utoledo.edu/human_chimp_macaque.html`). This triple alignment was used to calculate the dataset of recent mutations in humans. We considered a recent substitution at a particular position (for example T $\rightarrow$ C at position 23456719 on chromosome 7) to be valid if the human genome has a C base while both chimp and macaque have a T base in the corresponding aligned positions.

In addition, we required that the quality of the alignment in the vicinity of the mutation be reliable (more than 70% similarity between human and macaque in the 20 bp flanking region [-10, +10]). The frequency table of all inferred recent human mutations is presented in the Supplementary Table 2.6. We analyzed these recent substitutions together with the SNP datasets and call the former mutations "fixed substitutions," assuming that the majority of them occurred less than 10 million years ago and were already fixed across all human populations. In the same manner we processed indels in the triple alignments and computed all unambiguous cases of human insertions and deletions with sizes from 1 to 49 nucleotides.

2.5.2 Processing of SNP data.

Over 4.62 million human SNPs from all chromosomes were obtained (dbSNP build 128 [22], `ftp://ftp.ncbi.nih.gov/snp/`), filtered for completeness and correctness annotations (676499 records discarded total), and mapped onto the whole-genome human-chimpanzee alignment. SNP allele frequencies were averaged from the frequency data of all populations of that allele. However, only those SNPs that were successfully located within the alignment were processed further. For each SNP site we verified the existence of the particular polymorphic bases in the specified position of the human genome reference sequence and also in the corresponding aligned position on the chimp genomic sequence. If any of these two species had different bases than the SNP alleles, the SNP was discarded (20469 SNPs discarded total). Otherwise, we defined the origin of the polymorphism based on the chimpanzee nucleotide. Consider the following example to illustrate this process: suppose one has an A/G polymorphism located at position 34567812 of chromosome 5 with an average A allele frequency of 0.6 and a G allele frequency of 0.4. Then at position 34567812 of chromosome 5 of the human genome reference sequence (Genbank build 36.1), we would first examine if the A or G allele is present at that position and discard the SNP if not. Next, using the flanking region of that SNP we could align the chimp genomic sequence. If the chimp nucleotide were T or C then the SNP would also be discarded because those alleles are not a part of the human haplotype at that position. However supposing that the chimp nucleotide were G, then the polymorphism would be declared as a G→A polymorphism with G being declared the ancestral allele that at some point in human evolution mutated into an A allele within some human population(s). From the frequency data we may finally characterize this example SNP more precisely as a 0.4G → 0.6A polymorphism.

Using this approach we successfully characterized 3.93 million human SNPs. This last group of SNPs was divided into four subgroups based on the abundance of the mutant allele in the given human populations:

I. *rare polymorphisms with the frequency of the mutated allele being less than 3%;*

II. *minor polymorphisms with frequencies ranging from 3% to 20%;*

III. *medium polymorphisms with frequencies going from 20% to 80%; and*

IV. *major polymorphisms with the frequency being above 80%.*

For our method, misidentification of the ancestral allele might arise when the site for the human SNP is also polymorphic in chimp populations (e.g. A/G polymorphism) or for the possible case that this site had a recent substitution in chimps (A→G) after their divergence from humans. Human and chimpanzee genomes only differ by 1.23% due to single nucleotide substitutions with 1.06% being due to fixed substitutions and the rest (0.17%) being due to polymorphisms in human and chimp [20]. Moreover, according to the Chimpanzee Sequencing and Analysis Consortium the average estimated error rate of human alleles being misidentified due to chimp polymorphisms is only ∼1.6% across all typical SNPs, which is acceptably low. It is also observed, however, that in the mutational hotspot of the CpG dinucleotide, there is an increased error rate for ancestral misidentification. If the human alleles are C/T in the CpG and TpG context and the chimp allele is T (in the TpG context) then the estimated error rate is actually 9.8% [20]. Thus, in the context of studying our MRI regions, any substitution (especially in GC-rich MRI regions since they contain an overabundance of G and C) going from TpG $\rightarrow$ CpG could have the ancestral allele misidentified, which would mean that the substitution would actually be CpG $\rightarrow$ TpG, although in the case of GC-rich MRI regions where such dinucleotides are more likely, an error rate of 9.8% is not sufficient to change the trend or conclusion of our results.

2.5.3 X-rich MRI genomic regions and control regions with average base composition.

Any base or combination of bases can be described by a parameter X. For example, X could be G-base; C+T-bases; or A+T+G bases, et cetera. It is also useful to refer to *nonX* base(s) as all bases not X. Thus, $X + nonX$ must represent all four nucleotides A, G, T, and C. For the examples above, *nonX* are A+T+C-bases; G+A-bases, and C-base, respectively. MRI is characterized by a specific base composition within a region under analysis. We characterize X-rich MRI regions based on an overabundance of the X base(s) within a region of a certain length (the so-called window), where the percentage of X should be above a certain threshold (Bechtel et al 2008). We calculated MRI regions in the human genome for single nucleotides and various nucleotide combinations using a stretchy window of 100+ nucleotides with the following threshold parameters: for A or T the threshold was 49%; for G or C we used 40%; for G+C it was 70%; for A+T the threshold was at 80%; for G+T, C+A, G+A, and C+T were at 70%; nonA or nonT was 87%; and non G or non C the threshold was 93%. These thresholds were chosen experimentally in such a way that MRI regions should represent about 2% of the whole human genome. A stretchy window of $N+$ nucleotides means that we scan genomic sequence with an

N-size window to find a genomic MRI region that fits the threshold criterion, then we extend the window above the detected region by 10 nt steps until the criterion is no longer met. After registering the full MRI region we jump beyond the current MRI region and continue with the default N-size window. Using this approach we characterized all MRI regions in the triple human-chimp-macaque alignments using the human sequence for calculating nucleotide composition and MRI features. We also discarded those MRI regions in the alignments where the indel composition exceeded 50%. For the collection of control regions with average base compositions we used the same stretchy window approach with the nucleotide composition corresponding to the following average genomic frequencies: for A, T between 30 and 31% thresholds; for G, C between 20-21%; for G+C between 40-42%; A+T at 58-60%; G+T, C+A, G+A, or C+T were at 49-51%.

Note that control regions with genome-average X-composition also have genome-averaged $nonX$-composition. Therefore, their subsitution ratios are in inverse proportion to each other: $S_X = 1/S_{nonX}$. Due to this only one ratio for X and $nonX$ pair is shown in Figures 2-1 and 2-2.

2.5.4 Calculation of the substitution ratios in MRI and control regions.

Studying SNPs and fixed substitutions in X-rich MRI regions we measured the number of changes from X to $nonX$ (denoted as $N_{X \to nonX}$) and also the number of changes from $nonX$ to X (denoted as $N_{nonX \to X}$). The fluctuations in the observed distribution of $N_{X \to nonX}$ and $N_{nonX \to X}$ are well-known as Poisson noise. Thus, the standard deviation for the true values for $N_{X \to nonX}$ and $N_{nonX \to X}$ is calculated according to the Poisson distribution, that is: $\sigma_N = \sqrt{N}$. For each X-rich MRI region we measured the substitution S_X-ratios with $S_X = N_{X \to nonX}/N_{nonX \to X}$ shown in Figures 1 and 2. The propagation of uncertainty for a ratio $f = A/B$ can be calculated using the formula $(\sigma_f/f)^2 = (\sigma_A/A)^2 + (\sigma_B/B)^2 - 2(\sigma_A \cdot \sigma_B)/(A \cdot B) \cdot \rho_{AB}$, where ρ_{AB} is the correlation coefficient for A and B variables. Because the observed frequency of having a SNP at a genomic site in humans is less than 1%, it is correct to assume that the correlation between $N_{X \to nonX}$ and $N_{nonX \to X}$ is negligible. Therefore, the standard deviation for the S_X ratio was calculated by the following formula: $\sigma = (N_X/N_{nonX})\sqrt{(1/N_X) + (1/N_{nonX})}$.

2.5.5 Calculation of base composition equilibrium for the observed substitution rates.

As described in the previous paragraphs, for studying SNPs and fixed substitutions in X-rich MRI regions of the human genome we measured the number of changes from X to $nonX$ (denoted as $N_{X \to nonX}$) and also the number of changes from $nonX$ to X (denoted as $N_{nonX \to X}$). These $N_{X \to nonX}$ and $N_{nonX \to X}$ helped us to estimate the frequencies of these two types of mutations per X or $nonX$ site, named here as $F_{X \to nonX}$ and $F_{nonX \to X}$, correspondingly. Suppose one has a sample of MRI regions with a total nucleotide sequence length of L and a composition of X with the region richness given as P_X being measured in numbers from 0 to 1. Then, the total number of X sites in this sample will be $L \cdot P_X$, and the total number of $nonX$ sites will be $L \cdot (1 - P_X)$. During a certain time interval called ΔT there will be $\Delta N_{X \to nonX}$ and $\Delta N_{nonX \to X}$ substitutions. Therefore the frequency of substitutions per site is $F_{X \to nonX} = \Delta N_{X \to nonX}/(\Delta T \cdot L \cdot P_X)$ and $F_{nonX \to X} = \Delta N_{nonX \to X}/(\Delta T \cdot L \cdot (1 - P_X))$. It is impossible to measure directly these ΔN values for a specific time interval of ΔT. However, with a good approximation we can assume that the frequencies are proportional to the observed numbers $N_{X \to nonX}$ and $N_{nonX \to X}$ and can be represented by the simple formula: $F_{X \to nonX} = A \cdot N_{X \to nonX}/P_X$ and $F_{nonX \to X} = A \cdot N_{nonX \to X}/(1 - P_X)$, where A is a scaling factor having the same value for $F_{X \to nonX}$ and $F_{nonX \to X}$, since $N_{X \to nonX}$ and $N_{nonX \to X}$ are counted from the same sample. In a gedanken experiment, let's assume that the current $F_{X \to nonX}$ and $F_{nonX \to X}$ values will stay unchangeable forever for our MRI sample. Then, in time, mutations should alter the base composition of our sample until it reaches an equilibrium composition with a new percentage for X-bases denoted here as Q_X. This equilibrium composition Q_X can be computed using the observed parameters of P_X, $N_{X \to nonX}$, and $N_{nonX \to X}$. Indeed, under the equilibrium, the number of changes from X to $nonX$ must be equal to the number of reverse changes from $nonX$ to X, or:

$$\Delta N_{X \to nonX} = \Delta N_{nonX \to X} \tag{2.1}$$

We can compute these $\Delta N_{X \to nonX}$ and $\Delta N_{nonX \to X}$ values from frequencies in such a way:

$$\Delta N_{X \to nonX} = F_{X \to nonX} \cdot \Delta T \cdot L \cdot Q_X = A \cdot \frac{N_{X \to nonX}}{P_X} \cdot \Delta T \cdot L \cdot Q_X$$

also in a similar way

$$\Delta N_{nonX \to X} = F_{nonX \to X} \cdot \Delta T \cdot L \cdot (1 - Q_X) = A \cdot \frac{N_{nonX \to X}}{1 - P_X} \cdot \Delta T \cdot L \cdot (1 - Q_X)$$

By putting these transformations into Equation 2.1 we get:

$$A \cdot \Delta T \cdot L \cdot Q_X \cdot \frac{N_{X \to nonX}}{P_X} = A \cdot \Delta T \cdot L \cdot (1 - Q_X) \cdot \frac{N_{nonX \to X}}{(1 - P_X)}$$

or

$$Q_X \cdot \frac{N_{X \to nonX}}{P_X} = (1 - Q_X) \cdot \frac{N_{nonX \to X}}{(1 - P_X)} \tag{2.2}$$

Finally, simple transformation of Equation 2.2 gives us the final Equation 2.3 for calculation of equilibrium percentage:

$$Q_X = \frac{P_X \cdot N_{nonX \to X}}{N_{X \to nonX} \cdot (1 - P_X) + N_{nonX \to X} \cdot P_X} \tag{2.3}$$

In the Results section, Formula 2.3 is used to compute the equilibrium percentage for X-bases in the studied MRI regions.

2.6 Authors' contributions

AP, BRH, SPA, MC, JH, OMB were responsible for computational processing of the human-chimp-macaque datasets and creating the described programs. SS was responsible for the procuring and processing of the SNP data from dbSNP. JMB was responsible for the quantification of SNP data in the alignment. AP was also responsible for the processing of fixed substitutions and indel data from the three way alignment. AF supervised the project, provided guidance and wrote the draft. SS and AP also contributed to editing, typesetting, and writing the draft. All authors have read and approved the final manuscript.

2.7 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0643542. We thank Peter Bazeley, University of Toledo, for his computational support and discussion of our algorithms.

2.A Supplementary Tables

Table 2.4: Complete set of data representing the number of occurrences of major, middle, minor and rare SNPs in all X MRI regions (where X could represent either a two-base combination [GC, AG, GT, etc.] or a single base) and their corresponding control regions, which have an average nucleotide composition for the respective X.

All SNP occurrences in MRI and control regions.			
GC MRI REGIONS	GC Rich	non GC Rich	GC Average
Avg GC content (%)	71.3%	20.0%	40.7%
major SNPs ([AT]→[GC]):	365	688	151404
major SNPs ([GC]→[AT]):	700	455	123557
middle SNPs ([AT]→[GC]):	1197	3303	658643
middle SNPs ([GC]→[AT]):	4493	2148	707090
minor SNPs ([AT]→[GC]):	627	2431	452046
minor SNPs ([GC]→[AT]):	3310	1469	533913
veryrare SNPs ([AT]→[GC]):	483	1429	252603
veryrare SNPs ([GC]→[AT]):	3294	824	323430
GT MRI REGIONS	GT Rich	non GT Rich	GT Average
Avg GT content (%)	69.8%	30.1%	50.0%
major SNPs ([AC]→[GT]):	325	257	135693
major SNPs ([GT]→[AC]):	313	267	136029
middle SNPs ([AC]→[GT]):	1419	1490	678897
middle SNPs ([GT]→[AC]):	1812	1111	677630
minor SNPs ([AC]→[GT]):	851	994	488924
minor SNPs ([GT]→[AC]):	1287	703	489337
veryrare SNPs ([AC]→[GT]):	440	592	286952
veryrare SNPs ([GT]→[AC]):	779	354	286170
GA MRI REGIONS	GA Rich	non GA Rich	GA Average
Avg GA content (%)	70.0%	30.1%	50.0%
major SNPs ([TC]→[AG]):	366	387	45846
major SNPs ([AG]→[TC]):	497	358	45741
middle SNPs ([TC]→[AG]):	1939	2619	255252
middle SNPs ([AG]→[TC]):	3014	1745	255122
minor SNPs ([TC]→[AG]):	1199	1880	183317
minor SNPs ([AG]→[TC]):	2142	1107	183928
veryrare SNPs ([TC]→[AG]):	667	1140	106831
veryrare SNPs ([AG]→[TC]):	1192	607	106482

Continued on next page

Table 2.4: (continued)

All SNP occurrences in MRI and control regions.			
A MRI REGIONS	A Rich	non A Rich	A Average
Avg A content (%)	49.5%	12.9%	29.4%
major SNPs ([TCG]→[A]):	1364	841	70117
major SNPs ([A]→[TCG]):	1736	1017	83229
middle SNPs ([TCG]→[A]):	7077	5931	397843
middle SNPs ([A]→[TCG]):	9575	3349	373254
minor SNPs ([TCG]→[A]):	4640	4666	298658
minor SNPs ([A]→[TCG]):	6744	1936	257530
veryrare SNPs ([TCG]→[A]):	2549	3364	179847
veryrare SNPs ([A]→[TCG]):	3541	1044	144782
T MRI REGIONS	T Rich	non T Rich	T Average
Avg T content (%)	49.5%	12.9%	29.4%
major SNPs ([ACG]→[T]):	1431	831	69916
major SNPs ([T]→[ACG]):	1846	946	83452
middle SNPs ([ACG]→[T]):	7375	5541	398796
middle SNPs ([T]→[ACG]):	9666	3298	372713
minor SNPs ([ACG]→[T]):	4727	4463	299101
minor SNPs ([T]→[ACG]):	7093	1764	257460
veryrare SNPs ([ACG]→[T]):	2579	3202	180084
veryrare SNPs ([T]→[ACG]):	3879	947	144286
G MRI REGIONS	G Rich	non G Rich	G Average
Avg G content (%)	39.8%	7.0%	20.2%
major SNPs ([ATC]→[G]):	1583	909	86063
major SNPs ([G]→[ATC]):	1875	828	72169
middle SNPs ([ATC]→[G]):	5496	5535	393144
middle SNPs ([G]→[ATC]):	12741	3522	412683
minor SNPs ([ATC]→[G]):	2930	4383	271765
minor SNPs ([G]→[ATC]):	9632	2141	309842
veryrare SNPs ([ATC]→[G]):	1793	2529	153992
veryrare SNPs ([G]→[ATC]):	7045	1112	186220
C MRI REGIONS	C Rich	non C Rich	C Average
Avg C content (%)	39.9%	7.0%	20.2%
major SNPs ([ATG]→[C]):	1468	928	86391
major SNPs ([C]→[ATG]):	1911	800	72021
middle SNPs ([ATG]→[C]):	5740	5412	392481
middle SNPs ([C]→[ATG]):	12833	3570	413122
minor SNPs ([ATG]→[C]):	2919	4319	272763
minor SNPs ([C]→[ATG]):	9545	2237	309455
veryrare SNPs ([ATG]→[C]):	1708	2490	153323
veryrare SNPs ([C]→[ATG]):	7120	1201	186272

Table 2.5: Complete set of data representing the total number of X nucleotides (where X represents an X-rich MRI region) inserted or deleted due to short indels, whose sizes are less than three nucleotides in length, or medium indels, whose sizes range from three to fifty nucleotides in length, for all X MRI regions, and for the control regions, which have an average nucleotide composition for the respective X.

Indel data for X-rich MRI regions.			
GC MRI REGIONS			
MULTIPOINT INDEL (> 3 gaps)	GC Rich	non GC Rich	GC Average
Avg GC content (%)	71.5%	20.3%	40.9%
deletion A+T	4450	70651	220752
deletion G+C	11740	15065	108783
insertion A+T	3305	56595	119071
insertion G+C	11809	5386	62126
ΔGC	69	-9679	-46657
ΔAT	-1145	-14056	-101681
SHORT INDELS (< 2 gaps)	GC Rich	non GC Rich	GC Average
Avg GC content (%)	71.5%	20.3%	40.9%
deletion A+T	1062	21334	80780
deletion G+C	2254	5168	42376
insertion A+T	380	8105	45403
insertion G+C	1336	579	14995
ΔGC	-918	-4589	-27381
ΔAT	-682	-13229	-35377
GT MRI REGIONS			
MULTIPOINT INDEL (> 3 gaps)	GT Rich	non GT Rich	GT Average
Avg AC content (%)	69.1%	30.9%	50.0%
deletion A+C	10998	35900	206717
deletion G+T	41241	11166	208824
insertion A+C	5448	36577	119639
insertion G+T	40662	5783	120551
ΔAC	-5550	677	-87078
ΔGT	-579	-5383	-88273

Continued on next page

Table 2.5: (continued)

Indel data for X-rich MRI regions.			
SHORT INDELS (< 2 gaps)	GT Rich	non GT Rich	GT Average
Avg AC content (%)	69.1%	30.9%	50.0%
deletion A+C	3677	9368	77721
deletion G+T	10722	3671	78974
insertion A+C	725	4173	44671
insertion G+T	4464	776	45603
ΔAC	-2952	-5195	-33050
ΔGT	-6258	-2895	-33371
GA MRI REGIONS			
MULTIPOINT INDEL (> 3 gaps)	GA Rich	non GA Rich	GA Average
Avg GA content (%)	69.7%	30.2%	50.0%
deletion G+A	53068	17673	178435
deletion C+T	17816	56619	181165
insertion G+A	38369	7476	108860
insertion C+T	7514	38033	109420
ΔGA	-14699	-10197	-69575
ΔCT	-10302	-18586	-71745
SHORT INDELS (< 2 gaps)	GA Rich	non GA Rich	GA Average
Avg GA content (%)	69.7%	30.2%	50.0%
deletion G+A	15555	5050	65491
deletion C+T	5146	16730	66608
insertion G+A	6613	1312	38449
insertion C+T	1263	6836	38340
ΔGA	-8942	-3738	-27042
ΔCT	-3883	-9894	-28268
A MRI REGIONS			
MULTIPOINT INDEL (> 3 gaps)	A Rich	non A Rich	A Average
Avg A content (%)	49.6%	12.9%	30.5%
deletion A	45875	8626	84316
deletion G+C+T {non A}	32994	74011	177601
insertion A	36192	2884	50468
insertion G+C+T {non A}	17430	55500	109600
ΔA	-9683	-5742	-33848
Δnon A	-15564	-18511	-68001

Continued on next page

Table 2.5: (continued)

Indel data for X-rich MRI regions.			
SHORT INDELS (< 2 gaps)	A Rich	non A Rich	A Average
Avg A content (%)	49.6%	12.9%	30.5%
deletion A	14896	2253	32812
deletion G+C+T {non A}	11568	17473	70881
insertion A	7729	605	22478
insertion G+C+T {non A}	2384	6727	40113
ΔA	-7167	-1648	-10334
Δnon A	-9184	-10746	-30768
T MRI REGIONS			
MULTIPOINT INDEL (> 3 gaps)	T Rich	non T Rich	T Average
Avg T content (%)	49.5%	13.1%	30.5%
deletion T	52337	8406	86286
deletion G+C+A {non T}	32275	67314	174050
insertion T	39450	2943	49703
insertion G+C+A {non T}	17119	54071	107110
ΔT	-12887	-5463	-36583
Δnon T	-15156	-13243	-66940
SHORT INDELS (< 2 gaps)	T Rich	non T Rich	T Average
Avg T content (%)	49.5%	13.1%	30.5%
deletion T	17339	2201	33569
deletion G+C+A {non T}	11402	15770	69795
insertion T	8377	586	22914
insertion G+C+A {non T}	2474	6297	39678
ΔT	-8962	-1615	-10655
Δnon T	-8928	-9473	-30117
G MRI REGIONS			
MULTIPOINT INDEL (> 3 gaps)	G Rich	non G Rich	G Average
Avg G content (%)	40.1%	7.2%	20.4%
deletion G	17057	6682	51588
deletion T+C+A {non G}	26563	86293	264562
insertion G	17594	1346	32921
insertion T+C+A {non G}	17555	62525	164416
ΔG	537	-5336	-18667
Δnon G	-9008	-23768	-100146

Continued on next page

Table 2.5: (continued)

Indel data for X-rich MRI regions.			
SHORT INDELS (< 2 gaps)	G Rich	non G Rich	G Average
Avg G content (%)	40.1%	7.2%	20.4%
deletion G	4111	1914	20872
deletion T+C+A {non G}	6077	21845	101713
insertion G	2389	170	7759
insertion T+C+A {non G}	2221	8101	62733
ΔG	-1722	-1744	-13113
Δnon G	-3856	-13744	-38980
C MRI REGIONS			
MULTIPOINT INDEL (> 3 gaps)	C Rich	non C Rich	C Average
Avg C content (%)	40.1%	7.2%	20.5%
deletion C	16737	6391	52882
deletion T+G+A {non C}	26697	85129	264729
insertion C	17556	1394	31894
insertion T+G+A {non C}	18056	63772	161081
ΔC	819	-4997	-20988
Δnon C	-8641	-21357	-103648
SHORT INDELS (< 2 gaps)	C Rich	non C Rich	C Average
Avg C content (%)	40.1%	7.2%	20.5%
deletion C	3882	1868	20507
deletion T+G+A {non C}	6138	22006	99808
insertion C	2234	165	7672
insertion T+G+A {non C}	2361	8086	63125
ΔC	-1648	-1703	-12835
Δnon C	-3777	-13920	-36683

Table 2.6: Complete set of data representing the number of occurrences of fixed substitutions on all X MRI regions (where X could represent either a two-base combination [GC, AG, GT, etc.] or a single base), and the control regions, which have an average nucleotide composition for the respective X.

Dataset for fixed substitutions in X-rich MRI regions.			
GC MRI REGIONS	GC Rich	non GC Rich	GC Average
TOTAL No. AT→GC	28257	101790	1664017
TOTAL NO. GC→AT	44973	78722	1395170
TOTAL Neuclotides (L)	17878735	54775409	780583343

Continued on next page

Table 2.6: (continued)

Dataset for fixed substitutions in X-rich MRI regions.			
GT MRI REGIONS	GT Rich	non GT Rich	GT Average
TOTAL No. AC$\rightarrow$GT	63912	62631	2296863
TOTAL NO. GT$\rightarrow$AC	62605	62326	2299972
TOTAL Neuclotides (L)	34942524	34601253	1192711075
GA MRI REGIONS	GA Rich	non GA Rich	GA Average
TOTAL No. GA$\rightarrow$TC	55349	45921	719390
TOTAL NO. TC$\rightarrow$GA	45655	55511	721119
TOTAL Neuclotides (L)	69215018	69976866	978309614
A MRI REGIONS	A Rich	non G+A Rich	A Average
TOTAL No. A$\rightarrow$nonA	92225	68422	966600
TOTAL NO. nonA$\rightarrow$A	74211	66860	856488
TOTAL Neuclotides (L)	66902358	72373156	800413717
T MRI REGIONS	T Rich	non T Rich	T Average
TOTAL No. T$\rightarrow$nonT	92854	67100	961934
TOTAL NO. nonT$\rightarrow$T	76375	65495	858144
TOTAL Neuclotides (L)	67805198	71082740	800811487
G MRI REGIONS	G Rich	non G Rich	G Average
TOTAL No. G$\rightarrow$nonG	93059	43004	984588
TOTAL NO. nonG$\rightarrow$nonG	66086	48214	1124611
TOTAL Neuclotides (L)	52027684	60403314	884713535
C MRI REGIONS	C Rich	non C Rich	C Average
TOTAL No. C$\rightarrow$nonC	91642	42750	984160
TOTAL NO. nonC$\rightarrow$C	65945	48315	1119946
TOTAL Neuclotides (L)	51976831	60417430	883921526

References

[1] I. H. G. Consortium, "Finishing the euchromatic sequence of the human genome.," *Nature*, vol. 431, no. 7011, pp. 931–945, 2004 Oct 21.

[2] M. Suzuki and Y. Hayashizaki, "Mouse-centric comparative transcriptomics of protein coding and non-coding rnas.," *Bioessays*, vol. 26, pp. 833–843, Aug 2004.

[3] E. Segal, Y. Fondufe-Mittendorf, L. Chen, A. Thastrom, Y. Field, I. K. Moore, J.-P. Z. Wang, and J. Widom, "A genomic code for nucleosome positioning.," *Nature*, vol. 442, pp. 772–778, Aug 2006.

[4] S. Chattopadhyay and L. Pavithra, "Mars and marbps: key modulators of gene regulation and disease manifestation.," *Subcell Biochem*, vol. 41, pp. 213–230, 2007.

[5] S. Karlin and C. Burge, "Dinucleotide relative abundance extremes: a genomic signature.," *Trends Genet*, vol. 11, no. 7, pp. 283–290, 1995 Jul.

[6] M. Csuros, L. Noe, and G. Kucherov, "Reconsidering the significance of genomic word frequencies.," *Trends Genet*, vol. 23, pp. 543–546, Nov 2007.

[7] I. Rigoutsos, T. Huynh, K. Miranda, A. Tsirigos, A. McHardy, and D. Platt, "Short blocks from the noncoding parts of the human genome have instances within nearly all known genes and relate to biological processes.," *Proc Natl Acad Sci U S A*, vol. 103, pp. 6605–6610, Apr 2006.

[8] A. Meynert and E. Birney, "Picking pyknons out of the human genome.," *Cell*, vol. 125, pp. 836–838, Jun 2006.

[9] G. Bernardi, "The vertebrate genome: isochores and evolution.," *Mol Biol Evol*, vol. 10, no. 1, pp. 186–204, 1993 Jan.

[10] S. Karlin, A. M. Campbell, and J. Mrazek, "Comparative dna analysis across diverse genomes.," *Annu Rev Genet*, vol. 32, pp. 185–225, 1998.

[11] J. M. Bechtel, T. Wittenschlaeger, T. Dwyer, J. Song, S. Arunachalam, S. K. Ramakrishnan, S. Shepard, and A. Fedorov, "Genomic mid-range inhomogeneity correlates with an abundance of rna secondary structures.," *BMC Genomics*, vol. 9, p. 284, 2008.

[12] J. Mrazek and J. Kypr, "Middle-range clustering of nucleotides in genomes.," *Comput Appl Biosci*, vol. 11, pp. 195–199, Apr 1995.

[13] C. Nikolaou and Y. Almirantis, "A study of the middle-scale nucleotide clustering in dna sequences of various origin and functionality, by means of a method based on a modified standard deviation.," *J Theor Biol*, vol. 217, pp. 479–492, Aug 2002.

[14] R. Rudner, J. D. Karkas, and E. Chargaff, "Separation of b. subtilis dna into complementary strands. 3. direct analysis," *Proc Natl Acad Sci U S A*, vol. 60, pp. 921–2, Jul 1968.

[15] M. Kimura, *The Neutral theory of molecular evolution.* New York: Cambridge University Press, 1983.

[16] M. Gardiner-Garden and M. Frommer, "Cpg islands in vertebrate genomes.," *J Mol Biol*, vol. 196, pp. 261–282, Jul 1987.

[17] D. Takai and P. A. Jones, "The cpg island searcher: a new www resource.," *In Silico Biol*, vol. 3, no. 3, pp. 235–240, 2003.

[18] M. T. Webster and N. G. C. Smith, "Fixation biases affecting human snps.," *Trends Genet*, vol. 20, pp. 122–126, Mar 2004.

[19] L. Duret, A. Eyre-Walker, and N. Galtier, “A new perspective on isochore evolution.,” *Gene*, vol. 385, pp. 71–74, Dec 2006.

[20] C. Sequencing and A. Consortium, “Initial sequence of the chimpanzee genome and comparison with the human genome.,” *Nature*, vol. 437, no. 7055, pp. 69–87, 2005 Sep 1.

[21] R. Kuhn, D. Karolchik, A. Zweig, T. Wang, K. Smith, K. Rosenbloom, B. Rhead, B. Raney, A. Pohl, M. Pheasant, L. Meyer, F. Hsu, A. Hinrichs, R. Harte, B. Giardine, P. Fujita, M. Diekhans, T. Dreszer, H. Clawson, G. Barber, D. Haussler, and W. Kent, “The ucsc genome browser database: update 2009.,” *Nucleic Acids Res*, 2008 Nov 7.

[22] S. T. Sherry, M. H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin, “dbsnp: the ncbi database of genetic variation.,” *Nucleic Acids Res*, vol. 29, no. 1, pp. 308–311, 2001 Jan 1.

Chapter 3

The peculiarities of large intron splicing in animals

Authors:

Samuel S. Shepard[1,†], Mark McCreary[1,2,†],

& Alexei Fedorov[1,+]

†Equal contribution
+Corresponding author.

[1]Department of Medicine, University of Toledo, Health Science Campus, Toledo, OH, USA.
[2]Department of Biological Sciences, Rochester Institute of Technology, Rochester, NY, USA.

(Published in *PLoS One.* 2009 Nov 16;4(11):e7853.)

Abstract

In mammals a considerable 92% of genes contain introns, with hundreds and hundreds of these introns reaching the incredible size of over 50,000 nucleotides. These "large introns" must be spliced out of the pre-mRNA in a timely fashion, which involves bringing together distant $5'$ and $3'$ acceptor and donor splice sites. In invertebrates, especially Drosophila, it has been shown that larger introns can be spliced efficiently through a process known as recursive splicing—a consecutive splicing from the $5'$-end at a series of combined donor-acceptor splice sites called RP-sites. Using a computational analysis of the genomic sequences, we show that vertebrates lack the proper enrichment of RP-sites in their large introns, and, therefore, require some other method to aid splicing. We analyzed over 15,000 non-redundant, large introns from six mammals, 1,600 from chicken and zebrafish, and 560 non-redundant large introns from five invertebrates. Our bioinformatic investigation demonstrates that, unlike the studied invertebrates, the studied vertebrate genomes contain consistently abundant amounts of direct and complementary strand interspersed repetitive elements (mainly SINEs and LINEs) that may form stems with each other in large introns. This examination showed that predicted stems are indeed abundant and stable in the large introns of mammals. We hypothesize that such stems with long loops within large introns allow intron splice sites to find each other more quickly by folding the intronic RNA upon itself at smaller intervals and, thus, reducing the distance between donor and acceptor sites.

3.1 Introduction

Introns are found ubiquitously in eukaryotic genomes and yet their role is still poorly understood and underappreciated. A range of recent studies have suggested that introns may have even existed in what some regard to be primordial eukaryotes

[1–4] or even earlier [5,6]. Different aspects of the evolution of introns have been well reviewed by [7–9].

About 92% of mammalian genes have exon/intron structures while only 8% of genes are intron-free. The average segmented gene of these species contain between 8 and 9 introns. The total length of introns represents 35-40% of the euchromatic portion of mammalian genomes. Many introns are extremely long. For example, there are over 3000 human introns larger than 50 kb, 1,234 longer than 100 kb, 299 longer than 200 kb, and 9 longer than 500 kb [10]. The enormous size of introns in mammals creates several drawbacks. First, large introns waste considerable amounts of energy during transcription that is "unwisely" spent on polymerizing the extra-long intronic segments of pre-mRNA molecules. Second, large introns delay obtaining protein products. Third, large introns allow for more potential errors in intron splicing since large introns contain numerous false splice sites (the so-called "pseudo-exons" [11]). It follows that some benefit must therefore be associated with introns to compensate for these costly disadvantages. Different constructive roles for introns are described in [10].

In particular, we concentrate on the problems that large introns ($>$ 50 kb) pose to their host genes. During the initial steps of splicing, the $5'$-terminus of an intron is brought close to the downstream $3'$-terminus by the spliceosome RNA-protein complex. This spatial formation allows the phosphodiester bond at the donor splice site to be attacked by the $2'$-OH group of an adenosine residue from a so-called "branch point" located just in front of the acceptor splice site (on average, about 30 bases upstream). The larger the intron, the more remote its ends are from each other. At first approximation, the difficulty of bringing an intron's termini together in our three-dimensional world is proportional to the cube of the intron's length. Therefore, for a large 100 kb intron, it is one million times harder to bring its ends together than for a medium-sized intron of 1 kb in length. In fact, a stretched 100 kb RNA molecule spreads out over a distance of 30 microns, which is larger than the size of mammalian nucleus (about 5 microns). Moreover, splicing of large introns already takes extra time because there are so many bases to transcribe in the first place. Indeed, it takes about 45 minutes for RNA polymerase II to transcribe a 100 kb gene region. Thus, there is a fundamental question: How do large introns manage to splice at all?

Hatton *et al.* [12] as well as Burnette *et al.* [13] showed that Drosophila large introns undergo a process called *recursive splicing*; that is, several pieces of the intron are spliced consecutively starting from the $5'$-end. According to Burnette and colleagues, recursive splicing is achieved using a combined donor-acceptor splice site called the "ratcheting point" (or RP-sites). These RP-sites have a consensus of $(y)_n ncag|gtaagt$, where the splice junction is shown as vertical bar. The consensus sequences of the donor and acceptor splice sites are $AG|gtaagt$ and $(y)_n ncag|GT$ respectively (exon terminal sequences are shown in upper case and intron sequences are given in lower case). It is possible therefore that RP-sites may perform both functions–serving as either the $3'$- or $5'$- splicing junction—in order to facilitate recursive splicing. In 1998, Hatton *et al.* [12] described the existence of recursive splicing in fruit fly by quantitative RT-PCR. Afterwards, using different experimental techniques (RT-PCR of intermediate splicing products; RT-PCR test for lariat structures of intermediate introns; and mutational and deletion analyses of RP-sites) Burnette *et al.* [13] characterized in detail a mechanism that subdivides large introns by recursive splicing at non-exonic elements and alternative exons. The authors showed that RP-sites are 20-times more abundant in large Drosophila introns compared to their complementary strands as well as compared to their short introns.

In 2006 Grellscheid and Smith [14] showed that a pseudo-exon (a sequence within an intron flanked by bona fide $5'$ and $3'$ splice sites) in the rat tropomyosin gene was in fact most likely an alternative exon whose inclusion would lead to non-sense mediated decay. They also showed in their study on rat tropomyosin that a $5'$ splice site followed the pseudo-exon $3'$ splice site. They named this arrangement a "zero-length exon," which is equivalent in its form to an RP-site. Thus, RP-sites may indeed exist in mammals as well, although, as the authors suggest in their discussion, the function of the zero-length exon in this particular case is not likely to be the same as the RP-sites used for the recursive splicing of long introns as in the studies of Burnette *et al.* [13] in Drosophila. Few other examples of RP-sites in mammals exist in the literature at this time. An alternative hypothesis of large mammalian intron splicing has been proposed but not tested in [15].

Since mammals as well as many non-mammalian vertebrates have many more large introns than Drosophila, it follows then that there ought to be some aid to the removal of large introns if these species do not rely upon recursive splicing. Thus, we have performed a large-scale bioinformatics analysis to understand the possible splicing mechanisms for large introns in various vertebrate species. In particular, we predicted the number of stem structures within large introns, hypothesizing that periodic hairpins with stable stems and large loops may be a possible mechanism for pre-mRNA folding which could aid splicing efficiency.

3.2 Results

3.2.1 Distribution of large introns

Table 3.1 gives the distribution of large introns (> 50 kb) in thirteen completely sequenced genomes of both vertebrates and invertebrates. The genome sequencing quality varies significantly from species to species. This is reflected in the second to last column of Table 3.1, showing the percentage of unspecified nucleotides (non- A, T, C, or G) in the investigated large introns. Observe that some species (rat, cow, and sea urchin) have around 9% of their bases uncharacterized within large introns while other species (human, mouse, and fruit fly) have almost no uncharacterized bases. Even for the latter group of species there are different kinds of errors in the genomic databases, including sequencing, contig assembly, annotations, etc. (see the discussion in [16]). The reader should also note that some genes are still considered "hypothetical" and as such the counts in Table 3.1 are subject to future genome revisions.

Table 3.1: Large intron statistics and genome information by species. For columns left to right: number of non-redundant large introns (> 50 kb) in different animal genomes; genome size of each species; number of introns per gene; sequence quality of all large introns in the species (> 50 kb) as measured by percentage of ambiguous nucleotides (number of N's); sequence quality in the random set of large intron fragments used to predict stems.

Species	large introns (> 50 kb)	Genome size ($\times 10^9$ bp)	# introns per gene	large intron quality (% N's)	large intron fragment quality (% N's)
Human	3473	3.4	9.37	0.001	0.006
Mouse	2435	3.2	9.35	0.247	0.004
Rat	2332	3	9.17	8.442	0.672
Cow	2245	3.6	8.21	7.900	0.049
Dog	2223	3.4	9.79	0.572	0.004
Opossum	3270	3.5	8.88	1.495	0.028
Chicken	853	1.2	10.33	1.614	0.288
Zebrafish	756	1.9	8.29	4.926	0.892
Sea urchin	209	0.9	6.86	18.73	2.187
Fruit fly	45	0.2	3.98	0.000	0.004
Mosquito	7	0.27	3.3	0.122	0.154
Bee	199	0.19	6.2	1.578	0.199
Beetle	100	0.21	5	8.277	0.613

3.2.2 Distribution of splicing site motifs inside large introns

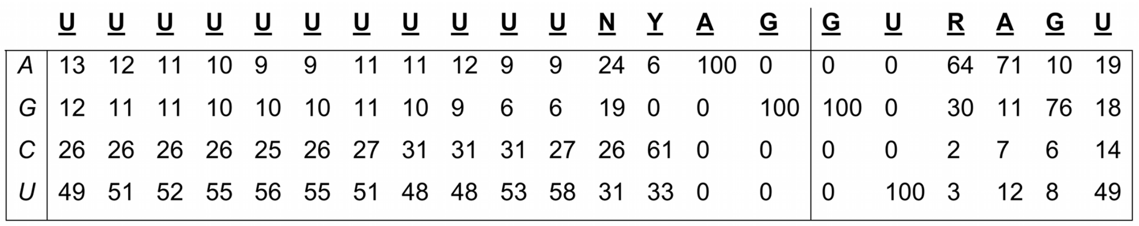
Table 3.2 shows the distribution of combined donor/acceptor sites for recursive splicing (RP-sites) within large introns and within their complementary strands. (For a study of RP-sites according to intron size class, see Supplementary Figure 3-5.) We scored RP-sites based on the human splice junction consensuses as shown in Figure

Table 3.2: Number of RP-sites per 100 kb inside large introns and their complementary sequences. The given ratio is the number of RP-sites of large introns to the number of RP-sites of the *complementary* sequences of the same large introns.

Species	Introns (> 50 kb)	Complementary Strands	RATIO (intr/comp)
Human	0.122	0.082	**1.5**
Mouse	0.087	0.078	**1.1**
Rat	0.078	0.074	**1.0**
Cow	0.112	0.098	**1.1**
Dog	0.139	0.107	**1.3**
Opossum	0.135	0.108	**1.2**
Chicken	0.105	0.102	**1.0**
Zebrafish	0.112	0.120	**0.9**
Sea urchin	0.540	0.066	**8.2**
Fruit fly	2.196	0.101	**21.7**
Mosquito	1.029	0.000	***>>10***
Bee	0.484	0.122	**4.0**
Beetle	0.807	0.101	**8.0**

3-1. However, very similar results were obtained when we used the splice junction consensuses of fruit fly, chicken, or zebrafish. Table 3.2 demonstrates that all of the studied invertebrates had a considerable enrichment of RP-sites within their large introns in contrast to their complementary sequences, which were used as the control. Contrary to this observation, mammals and other vertebrates had a much smaller abundance of RP-sites within their large introns compared to their complementary strands (a ratio of 1.5 or less). Supplementary Figure 3-5 demonstrates that RP-sites are many times more abundant in the larger introns of Drosophila than in its shorter introns, but in humans, intron size does not affect this enrichment. Additionally, Supplementary Figure 3-6 graphs the RP-site, 5′- and 3′- splice site enrichment ratios for all species. We also detected that mammalian and other vertebrate large introns had more stringent splice site motifs at their termini (the average score of large intron splice sites exceeded the average score of medium-sized introns by 10%).

In a similar manner, we calculated the distribution of donor and acceptor splice site motifs within the same set of large introns and their complementary strands



	U	U	U	U	U	U	U	U	U	U	U	N	Y	A	G	G	U	R	A	G	U
A	13	12	11	10	9	9	11	11	12	9	9	24	6	100	0	0	0	64	71	10	19
G	12	11	11	10	10	10	11	10	9	6	6	19	0	0	100	100	0	30	11	76	18
C	26	26	26	26	25	26	27	31	31	31	27	26	61	0	0	0	0	2	7	6	14
U	49	51	52	55	56	55	51	48	48	53	58	31	33	0	0	0	100	3	12	8	49

Figure 3-1: Ratcheting point consensus sequence (RP-site). The RP-site consensus was obtained from our purged sample of 11,315 non-redundant human gene sequences (with < 50% sequence identities between each other) from the human Exon-Intron Database, release 35p1. The top row contains the consensus sequence derived from the frequency information below. Each nucleotide in the consensus sequence is a column in the matrix whose rows show the frequencies found for each given nucleotide at that position. The first column gives the nucleotides corresponding to the frequency information.

(Tables 3.3 and 3.4 respectively). These computations were also based on the human splice junction consensuses with the assumption that 5′- and 3′-intron termini (GT or AG dinucleotides respectively) must be present in the RP-site motifs. These sites were counted when their scores exceeded eighty percent. It is clear from Tables 3.3 and 3.4 that the large introns of all studied species do not have any extreme excess of donor or acceptor splice sites compared to their complementary strands. This result stands in contrast to Table 3.2 and serves as a baseline for how often we should expect to find RP-sites on the complementary strand.

As an additional control, we used intergenic regions from human and fruit fly (see Methods) and measured the RP-site frequencies in those regions. Enrichment ratios of RP-sites for large introns versus their complementary sequences in human and fruit fly are 1.5 and 27.5 respectively. However, when we use intergenic regions as the control frequency, the RP-site ratios of large introns to intergenic regions are 1.3 and 29.7 for human and fruit fly respectively (see Supplementary Figure 3-7).

Table 3.3: Number of donor splice sites per 100 kb inside large introns and inside the complementary sequences of the same large introns.

Species	**Intron (> 50 kb)**	**Complementary Strands**
Human	34.102	35.257
Mouse	33.949	36.444
Rat	30.953	33.625
Cow	29.777	30.893
Dog	33.750	35.270
Opossum	37.091	39.522
Chicken	33.693	30.768
Zebrafish	24.940	25.740
Sea urchin	25.565	24.432
Fruit fly	19.836	22.235
Mosquito	20.792	24.292
Bee	14.722	17.225
Beetle	23.996	23.025

Table 3.4: Number of acceptor splice sites per 100 kb inside large introns and inside the complementary sequences of the same large introns.

Species	**Introns (> 50 kb)**	**Complementary Strands**
Human	10.012	7.764
Mouse	4.770	3.672
Rat	3.877	2.905
Cow	4.442	3.165
Dog	9.348	7.039
Opossum	4.417	3.540
Chicken	6.250	4.836
Zebrafish	5.078	4.694
Sea urchin	2.945	2.483
Fruit fly	4.900	3.548
Mosquito	3.706	1.235
Bee	5.458	5.191
Beetle	3.062	1.840

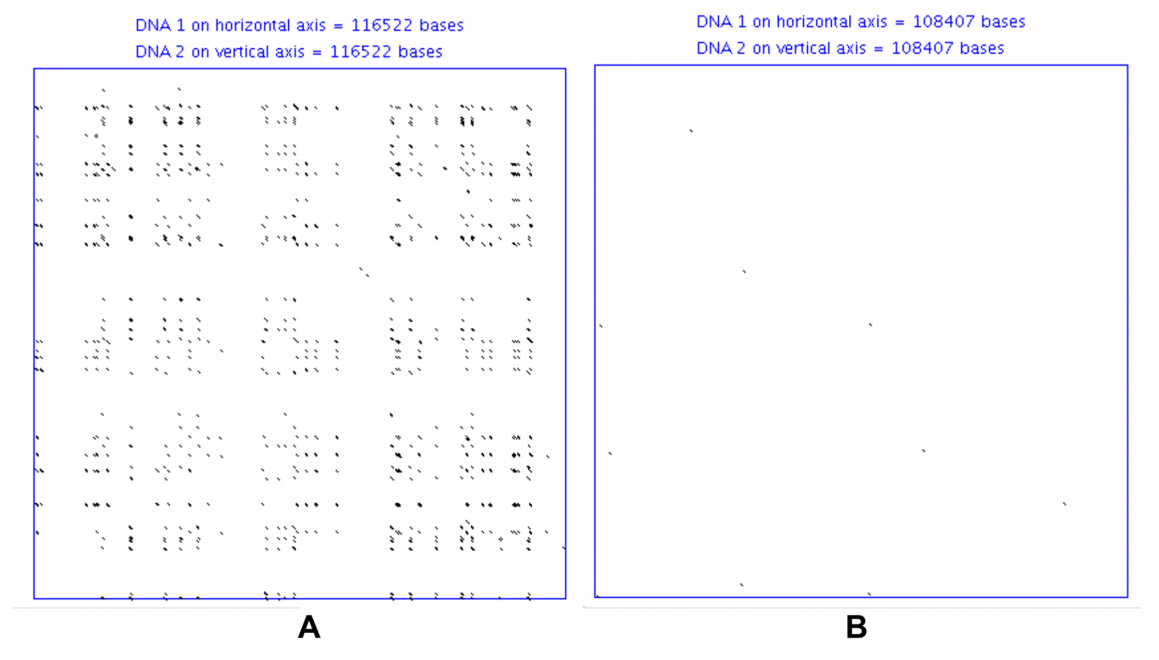


Figure 3-2: Human and Drosophila large intron dot-plots. **(A)** A dot-plot of human intron 21 from the *CNTNAP2* gene versus its complementary sequence. **(B)** Dot-plot of the drosophila intron 1 from the *luna* gene versus its complementary sequence. Here the dot-plot window size is 19 and the mismatch limit is set to 0. Low complexity repeats were filtered out using *RepeatMasker* before performing the dot-plot. The diagonal lines on the graph represent base pairing between different sections of the large introns that we may interpret as potential stem structures. The dot-plot conveys all possible combinations of stems in the sequence.

3.2.3 Searching for double-stranded secondary-structures inside large introns

RNA hairpin structures are crucial for the splicing of group I and group II introns [17,18]. A correlation between secondary structure of pre-mRNA spliceosomal introns and the efficiency of splicing has been described [19]. Hairpins inside spliceosomal introns can also regulate alternative splicing in many eukaryotic genes [20]. These facts give us the motivation to examine the abundance of possible hairpin structures in the large introns of vertebrates and to understand the role they might play in efficient splicing. Indeed, since vertebrates do not show an abundance of RP-sites we suppose that they must have some other mechanism for efficiently splicing large introns, which might be intron folding via multiple sequential hairpin structures. One of the simplest ways to visualize such hairpin structures is a dot-plot comparison of an intron sequence against its complementary strand, which is shown in Figure 3-2. Sequence segments that could form possible stem structures are plotted as short diagonal lines in this figure. Typical dot-plots for human and fruit fly large introns are given in Figures 3-2A and 3-2B respectively (human intron 21 of the *CNTNAP2* gene versus its complement and drosophila intron 1 of the *luna* gene versus its complement). Using *RepeatMasker* in this dot-plot analysis, we excluded all simple and low-complexity repeats (microsatellites, e.g. poly-AT sequences) from the analysis since they have an ability to interact with the nearest neighbor repetitive units rather than with more remote ones. In human, as with other mammals, the dot-plot detected a good number of matched segments throughout the entire large intron while the fruit fly showed very few possible stem structures. Examination of the predicted stem sequences of large introns showed that these possible stems are primarily formed by interspersed repeats belonging to the SINE and LINE classes. In the case of humans, the vast majority of the predicted stems are formed by any two oppositely oriented *Alu*-repeats, and, to a much lesser extent, *L1* or *L2* LINE repeats (see Table 3.5).

The direct computational method for the prediction of secondary structures in long RNA sequences, such as large introns, is not feasible because of the enormous sequence length [21]. Therefore, we first gathered potential stable stem structures indirectly by using BLAST alignments (and the dot-plots for visual inspection) of large intron sequences versus their complementary strand. Next, we applied the RNAcofold program to this loose dataset to actually predict stem structures—retaining all unique stems with an MFE -60 kcal/mol (see Table 3.6). We established that the actual choice of this threshold in the broad range of less than -50 to more than -100 kcal/mol has insignificant impact on the conclusions to the data. Thresholds higher than -50 kcal/mol represent much less stable structures. In the mammalian

Table 3.5: The DNA repeats associated with the predicted stems of Drosophila and human large intron fragments. The unique, predicted stems are from the same set of randomly selected large intron fragments from Table 3.6. Repeat families use *RepeatMasker* categories with the exception of "No Repeat Overlap", implying no such overlap was found between the strands of the predicted stems and repetitive elements, and "All Other Repeats" which is used to aggregate all other repeats less frequent than the "No Repeat Overlap" category. The average length of stem-repeat overlap for each repeat family is also given.

A	*Drosophila*		
Count	Percent	Average Overlap *Length* (bp) of Stem Sequences with Repeat Family	Repeat Family
2	100%	31	Simple Repeat

B	*Human*		
Count	Percent	Average Overlap *Length* (bp) of Stem Sequences with Repeat Family	Repeat Family
1534	81.7%	149	SINE/Alu
160	8.5%	259	LINE/L1
69	3.7%	118	LTR/MaLR
58	3.1%	39	Simple Repeats
26	*1.4%*	*0*	*No Repeat Overlap*
31	*1.9%*	*N/A*	*All Other Repeats*

pre-mRNA sequences there are a number of local structures of this strength and it is highly questionable that stable hairpins with thousands of nucleotides long loops could exist. The analysis revealed that almost all stable stems were formed by interspersed DNA repetitive elements in vertebrates and by simple repeats (except in the beetle) in invertebrates. Further examination of interspersed repeats in human large introns revealed that the human *Alu* repeats distributed with the same frequency in the (+) or (-) orientations and were randomly positioned along the intronic sequence. In short, we were unable to detect any pattern in the location and orientation of the repetitive elements compared to models where we randomly placed such elements along introns. It is also interesting to note that *Alu* elements were more common

Table 3.6: The features and frequencies of predicted stems for various species. Left to right we have the given species, the number of randomly selected large intron fragments (50 kilobases), the average number of stems per 50 kilobases, the average stem length, the average minimum free energy (MFE) of the stems, and the average loop size of the stems (in kilobases). All predicted stems were filtered to be less than or equal to -60 kcal/mol.

Species	*Number of 50 kb Intron Fragments*	*Stems per 50 kb*	*Avg. Stem Length (bp)*	*Avg. MFE of Stems (kcal/mol)*	*Average Loop Size (kb)*
Human	100	9.39	158	-258	12.3
Mouse	100	6.44	141	-229	13.5
Rat	100	5.54	156	-253	13.5
Cow	100	14.00	188	-310	14.4
Dog	100	8.02	112	-200	13.2
Opossum	100	5.73	138	-198	15.2
Chicken	100	1.36	95	-165	14.8
Zebrafish	100	8.72	114	-169	12.4
Sea Urchin	30	6.70	96	-142	10.8
Fruit Fly	30	0.03	32	-61	14.6
Mosquito	7	1.43	66	-114	12.6
Bee	30	0.53	56	-88	9.8
Beetle	30	4.00	155	-188	8.0

in human intergenic regions than human large introns and that, if transcribed, the number of predicted stems in intergenic regions would also be larger.

Our human intergenic region sample contained a total of 53.61% DNA repeats with 22.06% of the intergenic region being SINE compared to the sample of large introns which had 44.4% repeats with 12.36% SINE. For fruit fly, a 7.54% repeat composition in large introns jumped to 26.75% in intergenic regions (large retroviruses such as the ROO element appear in intergenic regions). Correspondingly, there were 19.56 unique, predicted stems per 50kb in human intergenic regions versus 9.39 in human large introns (-60 MFE). Drosophila had 1.16 predicted stems per 50kb in intergenic regions versus 0.03 in fruit fly large introns.

We detected negligible numbers of predicted stem structures formed by the ancient mammalian-wide interspersed repeats (MIR repeats from SINE class) presumably because they have accumulated too many mutations within each repetitive element to be adequately paired. Drosophila's only source of predicted stems came from simple repeats. See Table 3.5 for a full comparison of the human and drosophila DNA repeats that were associated with their respective predicted stems.

It is interesting to observe the sheer difference in magnitude in the number of stems between human and fruit fly. The average number of unique, predicted stem structures per 50 kb of large introns in different species is presented in Table 3.6. (We use the term "unique stem structures" to mean that any of the predicted stem's strands do not overlap with any other stem's sequences nor with each other.) Table 3.6 shows that these stems are about 1.4 to 420 times more abundant in mammalian large introns than in insects. The average lengths of these predicted stems are also given in Table 3.6, which shows that vertebrates only have at most 5.9 times the length of the stems found in invertebrates. However, from Table 3.6 it may also be argued that there is a trend for more stable stem structures in mammals and other vertebrates than in most invertebrates.

Apart from stems, we studied the composition of repetitive elements within large introns using the *RepeatMasker* program. The results are presented in Figure 3-3 and demonstrate that all of the studied invertebrates have no or negligible amounts of short or long interspersed repetitive elements, while mammals and non-mammalian vertebrates have the highest representation of these types of repeats. Of interest, the red-flour beetle (*Tribolium castaneum*) has the highest number of unique stems predicted for any insect as well as the most stable predicted stem structures for all insects. (Sea urchin, the only other invertebrate, has more predicted stems; however, they are primarily associated with simple repeats, see Discussion.) Strangely though, Figure 3-3 shows that beetle large introns have fewer repetitive elements than the rest of the studied invertebrate species. An example dot-plot for an entire beetle large intron is shown in Figure 3-4A while one example stem from the beetle stem

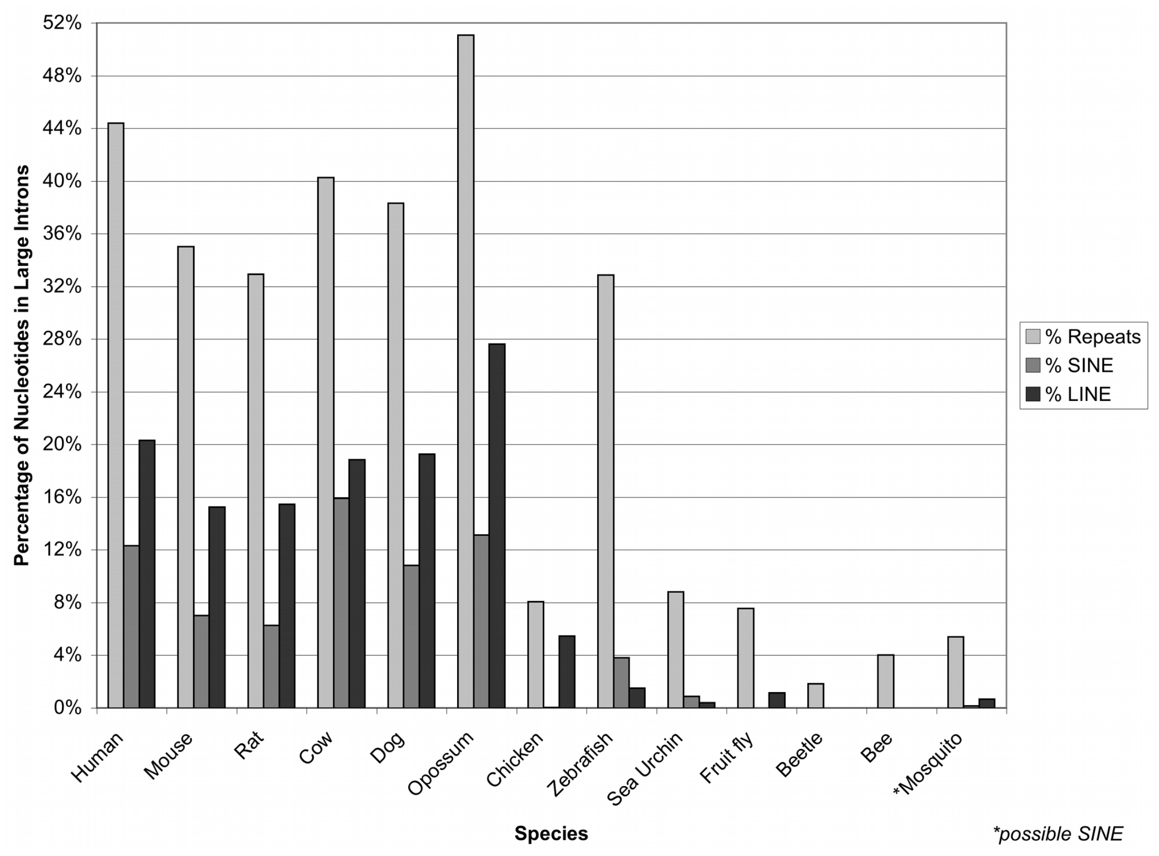


Figure 3-3: Repetitive elements within species. The percentage of repeats for the complete set of large introns for various species. The light gray bars are for the total percentage of repeats in large introns (percentage of nucleotides), the medium gray bars are only for the percentage of nucleotides made up by short interspersed element (SINE) repeats, while the dark gray bars are only for long interspersed element (LINE) repeats. **Note: Mosquito contains an ambiguous SINE element called "SINEX-1_AG."*

prediction is shown in Figure 3-4B. One may conclude that beetle large introns do possess a potential for stem structures that is unique among the studied insects, although these structures typically are not quite as abundant as the stems predicted for mammalian large introns. The repeat composition for the predicted stems of beetle large intron fragments (data not shown) reveals that over 90% of the stems are not associated with any known repeats.

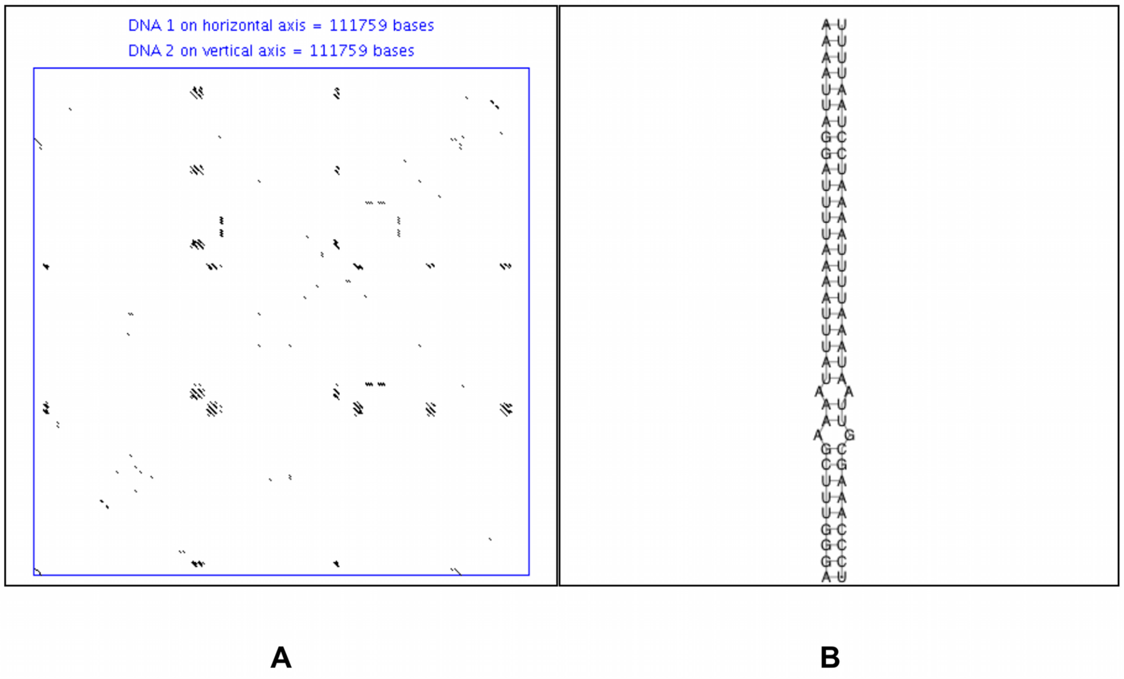

DNA 1 on horizontal axis = 111759 bases
DNA 2 on vertical axis = 111759 bases
A
B

Figure 3-4: Beetle large intron dot-plot and secondary structure. **(A)** Dot-plot of a beetle large second intron of the predicted gene XP_968205.1. The window size for the dot-plot was 19 and the mismatch limit was 0. **(B)** An example stem from the same intron, created using RNAcofold (it is not associated with any known repeat).

3.3 Materials and Methods

The sequences of non-redundant, large introns (> 50 kb) were obtained from the Exon-Intron Database [16]. Our datasets are available upon request. Supplementary Figure 3-5, Figure 3-3, and Tables 3.1–3.4 used these datasets. For the human intergenic region RP-site analysis we used the same data set as in [22], which contains over 3.5 million nucleotides. For the fruit fly intergenic region RP-site analysis we used the complete set of intergenic regions from FlyBase release 5.10 (`ftp://ftp.flybase.net/genomes/Drosophila_melanogaster/`) [23]. FlyBase release 5.10 was the same release used to build our Exon-Intron database from which we obtained the sample of large introns. See Supplementary Figure 3-7 for intergenic region RP-site analysis.

For the recognition of RP-sites we used the same computational algorithms as published by Burnette *et al.* 2005 with the same 80% scoring threshold for counting the number of RP-sites. In this computation we assumed that all RP-sites must have an invariable core sequence of AG|GT representing the intron's dinucleotide termini. The consensus for intron splicing junctions was obtained from our purged sample of 11,315 non-redundant human gene sequences (with $< 50\%$ sequence identities between each other) from the human Exon-Intron Database, release 35p1 [16]. Additionally, when comparing Drosophila with human introns in Supplementary Figure 3-5 we used the Drosophila consensus matrix to detect RP-sites in fruit fly and the human consensus matrix to detect RP-sites in *Homo sapiens.* Various scoring thresholds for human and Drosophila were used: 80%, 70%, and 60% with 80% being the highest quality RP-site recognition threshold. The intron size classes chosen for this analysis (1-6kb, 6-17kb, 17-41kb, 41-100kb, and 100+ kb) had a total of between 203 and 212 million bases for human. The fruit fly intron size classes were held to the same intervals allowing direct comparison of intron class size between Drosophila and human. For human and fruit fly RP-site analysis in intergenic regions we used the respective human and fruit fly consensus matrices.

The data used in the stem prediction and analysis (see Tables 3.5–3.6) was a randomly extracted set of large intron fragments from the datasets used in the RP-site analysis. We extracted 50,000 bp of fixed sequence fragments randomly from each of the large intron datasets of each species, but taking no more than one fragment from any particular large intron. For invertebrates (except mosquito), we randomly selected 45 fragments of 50 kb each and kept the 30 highest quality (by lowest number of N's) fragments. For mosquito, we randomly extracted 50 kb sequence fragments from each sequence in the mosquito large intron dataset (7 large intron fragments). For each vertebrate species we randomly selected 150 large intron fragments of 50 kb each and kept the 100 highest quality fragments. With respect to intergenic regions in human and fruit fly, we randomly extracted 100 fragments of 50 kb a piece from the respective datasets. The intergenic region fragments for human and fruit fly contained no ambiguous nucleotides. For a summary of the fragment quality of large introns, please see the last column of Table 3.1.

For the stem prediction, we initially gathered a rough pool of possible stems using blast2 alignments of large introns versus their complementary sequences. We used default parameters for *blastn* and matched only to the top or forward strand. From the blast2 alignments we actually predicted the stems using the RNAcofold program (Vienna package 1.6.1) with default parameters [24]. A custom perl program was used in concert with RNAcofold to: (a) retain structures with a minimum free energy (MFE) less than or equal to -60 kcal/mol; (b) discard palindromic structures (stems with no loops); (c) retain only unique stems; and (d) calculate statistics such as the average MFE, average stem length, and average loop size of the predicted stems. Predicted stems were considered *unique* if the stem's strands did not overlap with any other stem's sequences in the predicted stem set. Moreover, if a new stem to be added overlapped only one stem in the set and if the new stem had an MFE within 10% of the old stem and a smaller loop size, the algorithm would replace the old stem with the new one. The results are presented in Table 3.6.

Masking and characterization of repetitive elements inside introns (and intergenic regions) were performed with *RepeatMasker Open-3.1.8* (using the sensitive/slow search mode and species/genus specific repeat libraries [25]. The repeat libraries used by *RepeatMasker* was database *release 20061006* with *WUBlast 2.0MP* (to perform the scanning [26].

Cross-referencing between predicted stems sequences and RepeatMasker data for the large intron fragments of human and drosophila was performed using a custom perl program. The coordinates of the two sequences forming a predicted stem were each individually cross-referenced against the locations of all repeats in the respective 50 kb large intron fragment. For Table 3.5, if any overlap were found between the predicted stem sequences and a repeat it was counted. However, in order to verify the strength of the association between the repeats and the predicted stems, the lengths of their overlap for each repeat family was kept and the average length of the sequence overlap was calculated. The "No Repeat Overlap" category in Table 3.5 is not a RepeatMasker repeat family and has a 0 nucleotide average overlap length by definition. The "All Other Repeats" category in Table 3.5 are the aggregate count all other repeat families whose count is less than the "No Repeat Overlap" category. The average length of overlap is omitted for the "All Other Repeats" category since it contains many different repeat families each of a very low occurrence whose average is not reliably interpretable.

Dot-plot analysis was performed using a modified version of the Java applet "Nucleic Acid Dot Plot" [27]. The parameters used for Figures 3-2 and 3-4 included a window size of 19, a mismatch limit of 0, and masking of low-complexity repeats as X's.

All other computations were performed by programs written in perl and with queries performed in MySQL—all available upon request.

3.4 Discussion

The timely removal of large introns from pre-mRNA poses a challenging problem to spliceosomal machinery. It has been experimentally and computationally proven that in *Drosophila melanogaster* there exists a special strategy named recursive splicing for the excision of large introns. Recursive splicing occurs via selective accumulation of combined donor-acceptor splicing sites called RP-sites [12, 13]. In our research, we used the complete set of Drosophila large introns to confirm the previous computation by Burnette *et al.* [13]—showing again that fruit fly had more than 20 times the selective accumulation of RP-sites within large introns over their complementary strands. Similarly, all other studied Insecta species (mosquito, honey bee, and beetle) as well as more distant invertebrates (sea urchin) also had an accumulation of RP-sites within their large introns that was several times more abundant when compared to their complementary strand. We also showed that the accumulation of RP-sites is in particular with respect to intron class size in fruit fly but not in human (Supplementary Figure 3-5). On the other hand, all studied vertebrates, including six mammals, did not show significant accumulation of RP-sites (see Supplementary Figure 3-6 for a visual representation of this phenomena). Moreover, vertebrate species have overwhelmingly more large introns than the examined invertebrates. Therefore, vertebrates must mobilize another molecular mechanism for the removal of their large introns from pre-mRNA. We have hypothesized that multiple hairpins with large loops could form compact spatial structures within large introns that could help put the donor and acceptor splice sites in close proximity in order to facilitate splicing.

To test this conjecture, we examined the distribution of possible stable stem structures inside the large introns of vertebrates and invertebrates. It appeared that within Drosophila's large introns, stem structures are practically absent. The same trend was observed for the invertebrates honey bee and mosquito. On the other hand, in mammals, multiple SINE and LINE repeats (primarily SINE) located in different orientations throughout large introns drive the potential formation of hairpins with large loops. For humans there were an average of about 9.4 possible hairpins per 50 kb of the analyzed large intron sequence fragments. A vast majority of these possible stems are formed by oppositely oriented primate-specific *Alu*-repeats (81.7%). Other investigated mammals do not have *Alu*-elements, but other types of evolutionarily new SINEs specific for their taxa. These SINEs could also allow for the formation of multiple hairpin structures inside large introns. Only one of the studied vertebrates, chicken, does not have SINE elements in its genome. Instead, the chicken has very abundant and relatively short LINE elements that comprise over 60% of its repetitive elements. Thus in chicken large introns, possible stems may be formed solely by LINE repeats and not SINE repeats. One may observe, however, that the chicken has very few predicted stems, less than all studied vertebrate species and comparable to some insect species. It may be the fact that avian genomes deal with large intron splicing differently than other vertebrate species. Two facts though are clear: predicted stems for chicken are quite strong and stable (see Table 3.6) and the chicken has several times fewer large introns than all studied mammalian species (see Table 3.1).

Interestingly, the beetle and especially the sea urchin contain the most predicted stems of all studied invertebrates. While the sea urchin may contain the most predicted stems, even comparable to zebrafish, the majority of these predicted stems (47.5%) overlap with simple and low complexity repeats that might form hairpin structures without loops instead of the stems with large loops that we predict in mammals. Curiously, beetle's predicted stems are not strongly associated with any particular kind of repeat. We suppose that the beetle predicted stems might be formed by as yet unidentified repeats, or that they are merely a part of more complicated RNA secondary structures.

The average number of predicted long and stable stems in large introns of different mammals is 5.5 to 14 per 50 kb of large introns (see Table 6). These stems create large loops with the average size of 12.3 to 15.2 kilobases. Relatively large loops with lengths up to 3 kilobases are characteristic for group I and group II introns containing ORFs. According to [28], about 30% of group I introns and about 25% of group II introns code proteins. These coding sequences are located inside loops that do not have specific secondary structures. The ORF-containing loops of group I introns are around 1000 nucleotides in length, while those of group II introns are even larger. The latter code proteins with an average size of 500-600 aa, according to the Group II intron database [29]. Moreover, some of these proteins are significantly larger (up to 1064 aa in M.p.atpAI1 intron [29]). Interestingly, these large ORF-containing loops of group I and II introns have relatively short terminal stems, usually no longer than 12 nucleotides with MFE weaker than -10 kcal/mol (P6 or P8 stems for group I; IV stems for group II introns). Multiple hairpins of these introns form complex 3D structures. These complex 3D-structures include pseudoknots and non-Watson-Crick base pairing. Presently, there are no reliable algorithms/programs to properly calculate the free energy of such structures. Therefore we do not provide such estimations. However, each individual stem of group I and II introns has folding energy at least ten times weaker than -258 kcal/mol—the average minimum free energy of the predicted stems of large introns in human (see Table 3.6). Therefore, it is reasonable to hypothesize that numerous SINE and LINE repetitive elements within large mammalian introns are able to form multiple large hairpins with 100-300 nucleotide-long stems and up to a 15 kb long loops. Such structures might help to bring donor and acceptor splicing junctions of large introns closer to each other, and, thus, facilitate the effectiveness of their splicing. Indeed, recently it has been shown that even in the short introns of Saccharomyces cerevisiae secondary structures facilitate splicing by bringing together splicing elements [19].

Insertion of interspersed retrotransposon elements, such SINEs and LINEs, is a major force for the expansion of the genome size as a whole and intron sizes in particular [30]. Accumulation of new types of retrotransposons occurs gradually and could take millions of years. After gaining several interspersed repetitive elements inserted in opposite orientations inside an intron, these elements could allow for the formation of hairpin structures with long stems to be formed by the base-pairing repetitive sequences. These hairpins would introduce a new spatial organization into intronic RNA by keeping donor and acceptor splice sites in close proximity. Such a spatial organization could become a novel mechanism for facilitating the splicing of large introns. If RP-sites were indeed already present, this competing mechanism for efficient splicing could, in turn, ease the selective constraints that preserve recursive splicing and decrease RP-site frequency to a random expectation. We therefore hypothesize that oppositely oriented interspersed repetitive elements may be playing this role in the large introns of vertebrate species. It is indeed interesting to consider that the possible problems caused by the expansion of introns due to the insertion of repetitive elements may at once be remediated by the base-pairing of the self-same elements. However, whatever forces drove or allowed the formation of such possible stem structures, their potential role in the efficient splicing of large introns poses an appealing question to molecular biologists, a question that is suggestive for future work *in vitro*.

3.5 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0643542. We thank Valery Shepelev for discussion and sample preparation as well as Peter Bazeley for his help with the installation of our analysis tools.

3.A Supplementary Figures

Figure 3-5: RP-site enrichment with respect to intron size. Human and Drosophila RP-site enrichment ratio calculated for various scoring thresholds and intron size classes. The RP-site ratio is the count of RP-sites on the direct strand of introns divided by the count of RP-sites on the complementary strand of said introns. Thresholds for scoring or recognizing RP-sites to a consensus sequence are 80%, 70%, and 60% with 80% being the most stringent (good quality) score. Intron class sizes are the five sets with individual intron lengths: 1) 1–6 kb, 2) 6–17 kb; 3) 17–41 kb; 4) 41–100 kb; and 5) larger than 100 kb.

RP-site Enrichment in Intron Size Classes by Species and Scoring Threshold

		Human			**Fruit Fly**		
	≥score threshold	*80%*	*70%*	*60%*	*80%*	*70%*	*60%*
INTRON CLASS SIZE	1 to 6 kb	1.1	1.2	1.2	0	0.2	0.3
	6 to 17 kb	1.4	1.3	1.3	2.5	1.2	0.8
	17 to 41 kb	1.3	1.4	1.4	14.5	6.9	3
	41 to 100 kb	1.6	1.3	1.3	36	10	6.1
	100+ kb	1.5	1.3	1.3	67*	12	5.3

* Drosophila large intron group 100+ kb with scoring threshold 80% ratio is estimated, since 8 to 0 cannot be divided, using a polynomial curve fit ($R^2 = 1$) to the previous four points.

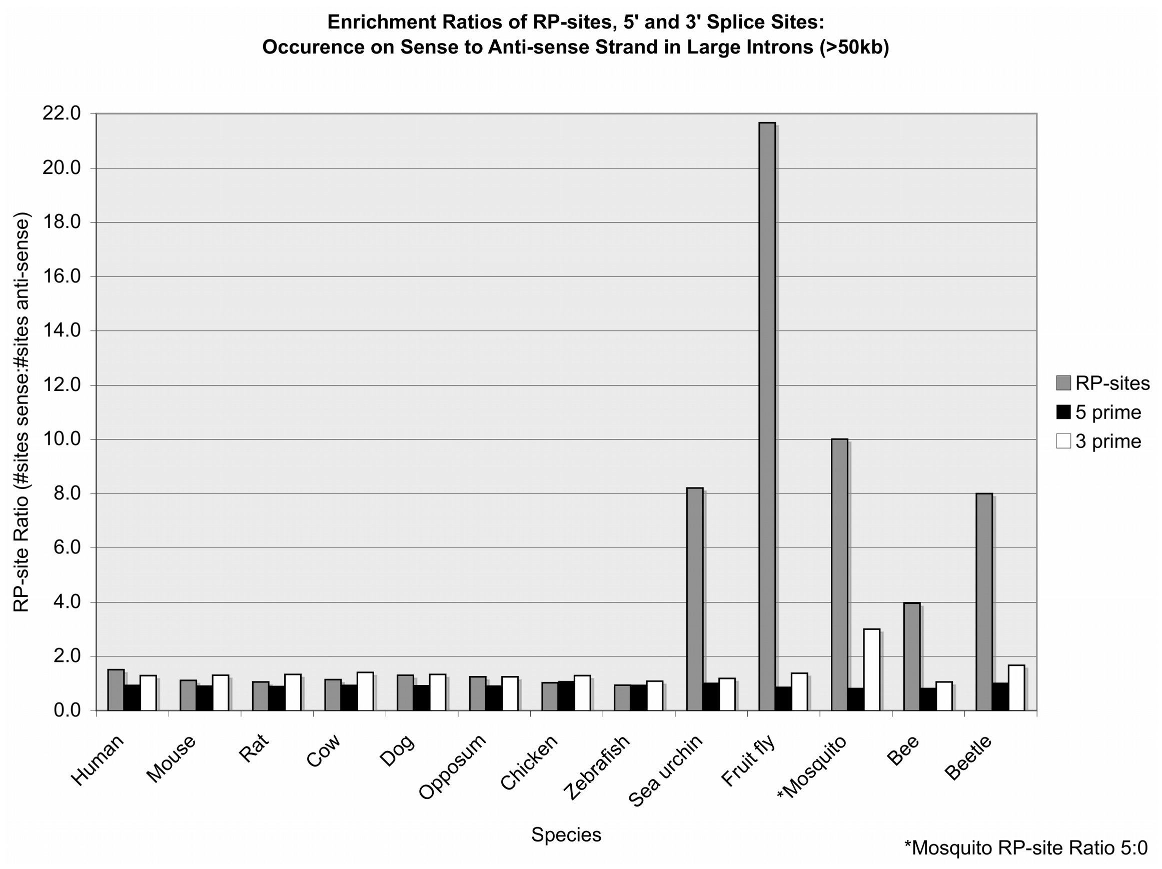


Figure 3-6: RP-site ratio comparison. In various species, the ratios of the number of sites (RP-site, 5-prime, or 3-prime) on the sense strand of large introns (> 50 kb) is compared to the number of sites on the anti-sense strand of large introns.

Figure 3-7: Comparison of controls used to calculate RP-site enrichment ratios in large introns. The complementary strand of the large introns is used as the first control. The RP-site enrichment ratio is the frequency of RP-sites in the direct strand of large introns to the frequency of RP-sites in the direct strand of large introns. A set of 50 kb intergenic region fragments is used as a second control. The RP-site enrichment ratio here is the frequency of RP-sites in the direct strand of large introns to the frequency of RP-sites in intergenic regions. Both human and fruit fly species are considered with RP-sites being calculated at the 80% scoring threshold. The human consensus matrix was used for human and the fruit fly consensus matrix was used for fruit fly. All frequencies are per 100 kilobases.

Complementary Strand of Large Introns as Control

	RP-sites per 100kb (Large Introns)	*RP-sites per 100kb (Complementary Strand)*	*RP-site Enrichment Ratio*
Human	0.122	0.082	1.5
Fruit Fly	1.859	0.068	27.5

Intergenic Region as Control

	RP-sites per 100kb (Large Introns)	*RP-sites per 100kb (Intergenic Region)*	*RP-site Enrichment Ratio*
Human	0.122	0.096	1.3
Fruit Fly	1.859	0.063	29.7

References

[1] R. Belshaw and D. Bensasson, "The rise and falls of introns," *Heredity*, vol. 96, pp. 208–13, Mar 2006.

[2] A. Fedorov, A. F. Merican, and W. Gilbert, "Large-scale comparison of intron positions among animal, plant, and fungal genes," *Proc Natl Acad Sci U S A*, vol. 99, pp. 16128–33, Dec 2002.

[3] I. B. Rogozin, Y. I. Wolf, A. V. Sorokin, B. G. Mirkin, and E. V. Koonin, "Remarkable interkingdom conservation of intron positions and massive, lineage-specific intron loss and gain in eukaryotic evolution," *Curr Biol*, vol. 13, pp. 1512–7, Sep 2003.

[4] S. W. Roy and W. Gilbert, "The evolution of spliceosomal introns: patterns, puzzles and progress," *Nat Rev Genet*, vol. 7, pp. 211–21, Mar 2006.

[5] S. J. de Souza, M. Long, R. J. Klein, S. Roy, S. Lin, and W. Gilbert, "Toward a resolution of the introns early/late debate: only phase zero introns are correlated with the structure of ancient proteins," *Proc Natl Acad Sci U S A*, vol. 95, pp. 5094–9, Apr 1998.

[6] A. Fedorov and L. Fedorova, "Introns: mighty elements from the rna world," *J Mol Evol*, vol. 59, pp. 718–21, Nov 2004.

[7] M. Lynch and A. O. Richardson, "The evolution of spliceosomal introns," *Curr Opin Genet Dev*, vol. 12, pp. 701–10, Dec 2002.

[8] L. Collins and D. Penny, "Complex spliceosomal organization ancestral to extant eukaryotes," *Mol Biol Evol*, vol. 22, pp. 1053–66, Apr 2005.

[9] I. B. Rogozin, A. V. Sverdlov, V. N. Babenko, and E. V. Koonin, "Analysis of

evolution of exon-intron structure of eukaryotic genes," *Brief Bioinform*, vol. 6, pp. 118–34, Jun 2005.

[10] L. Fedorova and A. Fedorov, "Puzzles of the human genome: Why do we need our introns?," *Current Genomics*, vol. 6, pp. 589–595, December 2005.

[11] H. Sun and L. A. Chasin, "Multiple splicing defects in an intronic false exon," *Mol Cell Biol*, vol. 20, pp. 6414–25, Sep 2000.

[12] A. R. Hatton, V. Subramaniam, and A. J. Lopez, "Generation of alternative ultrabithorax isoforms and stepwise removal of a large intron by resplicing at exon-exon junctions," *Mol Cell*, vol. 2, pp. 787–96, Dec 1998.

[13] J. M. Burnette, E. Miyamoto-Sato, M. A. Schaub, J. Conklin, and A. J. Lopez, "Subdivision of large introns in drosophila by recursive splicing at nonexonic elements," *Genetics*, vol. 170, pp. 661–74, Jun 2005.

[14] S.-N. Grellscheid and C. W. J. Smith, "An apparent pseudo-exon acts both as an alternative exon that leads to nonsense-mediated decay and as a zero-length exon," *Mol Cell Biol*, vol. 26, pp. 2237–46, Mar 2006.

[15] S. Ott, Y. Tamada, H. Bannai, K. Nakai, and S. Miyano, "Intrasplicing–analysis of long intron sequences," *Pac Symp Biocomput*, pp. 339–50, 2003.

[16] V. Shepelev and A. Fedorov, "Advances in the exon-intron database (eid)," *Brief Bioinform*, vol. 7, pp. 178–85, Jun 2006.

[17] A. M. Pyle, O. Fedorova, and C. Waldsich, "Folding of group ii introns: a model system for large, multidomain rnas?," *Trends Biochem Sci*, vol. 32, pp. 138–45, Mar 2007.

[18] Q. Vicens and T. R. Cech, "Atomic level architecture of group i introns revealed," *Trends Biochem Sci*, vol. 31, pp. 41–51, Jan 2006.

[19] S. Rogic, B. Montpetit, H. H. Hoos, A. K. Mackworth, B. F. Ouellette, and P. Hieter, “Correlation between the secondary structure of pre-mrna introns and the efficiency of splicing in saccharomyces cerevisiae,” *BMC Genomics*, vol. 9, p. 355, 2008.

[20] E. Buratti and F. E. Baralle, “Influence of rna secondary structure on the pre-mrna splicing process,” *Mol Cell Biol*, vol. 24, pp. 10505–14, Dec 2004.

[21] D. H. Mathews, “Predicting a set of minimal free energy rna secondary structures common to two sequences,” *Bioinformatics*, vol. 21, pp. 2246–53, May 2005.

[22] J. M. Bechtel, T. Wittenschlaeger, T. Dwyer, J. Song, S. Arunachalam, S. K. Ramakrishnan, S. S. Shepard, and A. Fedorov, “Genomic mid-range inhomogeneity correlates with an abundance of rna secondary structures,” *BMC Genomics*, vol. 9, p. 284, 2008.

[23] S. Tweedie, M. Ashburner, K. Falls, P. Leyland, P. McQuilton, S. Marygold, G. Millburn, D. Osumi-Sutherland, A. Schroeder, R. Seal, H. Zhang, and FlyBase Consortium, “Flybase: enhancing drosophila gene ontology annotations,” *Nucleic Acids Res*, vol. 37, pp. D555–9, Jan 2009.

[24] I. L. Hofacker, “Vienna rna secondary structure server,” *Nucleic Acids Res*, vol. 31, pp. 3429–31, Jul 2003.

[25] A. F. A. Smit, R. Hubley, and P. Green, “Repeatmasker open-3.0.” `http://www.repeatmasker.org`, 1996-2004.

[26] W. Gish, “Wublast 2.0mp.” `http://blast.wustl.edu`, 1996-2006.

[27] R. Bowen, “Nucleic acid dot plots.” `http://www.vivo.colostate.edu/molkit/dnadot/`, 1998.

[28] A. M. Lambowitz, M. Caprara, S. Zimmerly, and P. Perlman, "Group i and group ii ribozymes as rnps: Clues to the past and guides to the future.," in *The RNA World* (R. Gesteland, T. Cech, and J. Atkins, eds.), pp. 451–485, Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 2006.

[29] L. Dai, N. Toor, R. Olson, A. Keeping, and S. Zimmerly, "Database for mobile group ii introns," *Nucleic Acids Res*, vol. 31, pp. 424–6, Jan 2003.

[30] J. Brosius, "The contribution of rnas and retroposition to evolutionary novelties," *Genetica*, vol. 118, pp. 99–116, Jul 2003.

Chapter 4

Binary-abstracted Markov models and their application to sequence classification

Authors:

Samuel S. Shepard[1], Gursel Serpen[2],

Andrew McSweeny[3], and Alexei Fedorov[1,+]

+Corresponding author.

[1]Department of Medicine, University of Toledo, Health Science Campus, Toledo, OH, USA.
[2]Department of Electrical Engineering and Computer Science, University of Toledo, Toledo, OH, USA.
[3]Program in Bioinformatics and Proteomics/Genomics, University of Toledo, Health Science Campus, Toledo, OH USA.

Abstract

Markov models have been an indisputably useful tool in the field of bioinformatics. In the current study we extend and complement existing Markov model algorithms by developing and testing a novel binary-abstracted Markov model (BAMM) algorithm. BAMM can emphasize selected portions of genomic sequence signals according to specific abstraction rules. We present abstraction rules that generalize genomic sequence patterns at the single nucleotide level up to the level of tetranucleotides, using both in-frame data and data of mixed reading frames. Moreover, we show context-dependent abstraction rules that emphasize genomic duplication events as well as different kinds of sequence repetition. Unlike traditional Markov models, BAMM can analyze nucleotide patterns on the short-range (< 20 bp) up to the mid-range (20 to 50 bp) scale. Abstraction rules can also be both frame sensitive or independent. We build classifiers for both coding sequences and introns as well as for $5'$ and $3'$ UTR data. Finally, using support vector machines, we demonstrate that we can combine multiple BAMM classifiers, and indeed, traditional homogeneous Markov chains, to get even better exon-intron classification accuracy.

4.1 Introduction

Markov models (MM) are a popular technique in bioinformatics and gene prediction [1, 2]. More advanced (and popular) methods such as hidden Markov models [3] may still rely on the trained variables produced for inhomogeneous (reading frame-specific) Markov chains [4], so the core Markov chain technology is still relevant even today. Some of the latest gene prediction techniques can do "self-training" on sufficiently large genomes without the aid of already characterized training sequences; however, even these *ab initio* gene finders must still be instantiated with some initial model parameters (such as those of an inhomogeneous Markov model) in order to iteratively converge to the locally optimal gene finding model [5, 6]. Thus, new explorations in Markov chain technology may one day lead to their own evolution in gene finding approaches.

The *order* of a Markov model pertains to the "memory" of the Markov chain; that is, the number of nucleotides that are "remembered" by the algorithm. Generally, the order of the Markov model ranges from order 0 (no memory) to about 5 nucleotides [2, 7], with larger order Markov models almost always being *more* accurate. In order to train a Markov model of order 5, the researcher will need to analyze the frequencies of all 6-mers within the training sequences. This is quite feasible with a large training set of coding exons and introns, but longer nucleotide patterns ($>$ 7 bp) will be increasingly difficult to analyze with a Markov chain (in terms of order). We therefore wish to develop a new approach that can make use of nucleotide patterns longer than 7 base-pairs.

Indeed, there is a definite reason to be interested in nucleotide patterns longer than 7 bp. DNA at the genomic mid-range scale (around 30 to 1000 bp) contains a non-randomness or inhomogeneity that is associated with predicted strong, local RNA secondary structures [8]. Moreover, this mid-range inhomogeneity or MRI is genome-wide and appears to be maintained by some kind of selection pressure [9]. Unfortunately, traditional nucleotide Markov models are unsuitable for analyzing such sequence signals, as massive amounts of training data would be needed. In order to address this concern (as well as create the ability to emphasize particular nucleotide patterns) we have developed a novel approach called the binary-abstracted Markov model (BAMM) for sequence classification. BAMM's initial efficacy will be tested on the human genome, which has been well-characterized [10] and is highly relevant to researchers.

We present various "abstraction rules" (rules for emphasizing certain nucleotide patterns) for our binary-abstracted Markov models, implement individual BAMM sequence classifiers, and finally combine multiple BAMM classifiers under a machine-learning algorithm known as the support vector machine. Machine-learning is a useful tool to combine multiple sources of evidence in order to do classification. Support vector machines can be applied to biological data for splice site recognition [11] as well as gene finding (mGene) [12] with good results. Additionally, use of multiple gene predictions (via a machine-learning method known as a decision tree) has already been shown to produce better results than the individual gene predictions [13]. Because of our emphasis on middle-range nucleotide patterns, we believe our method may complement and enhance existing gene finding approaches as it matures, though it will probably never replace them (one can construct BAMM classifiers that are equivalent to traditional homogeneous Markov models but this will not usually be the case). In its current stage BAMM is presented as a simple sequence classifier for discriminating coding exons and introns as well as $5'$ untranslated regions. While we believe our results are sufficient to prove BAMM's promise as a sequence classification tool, further development must be undertaken before BAMM can become a full-fledged gene finding program of its own.

4.2 Methods

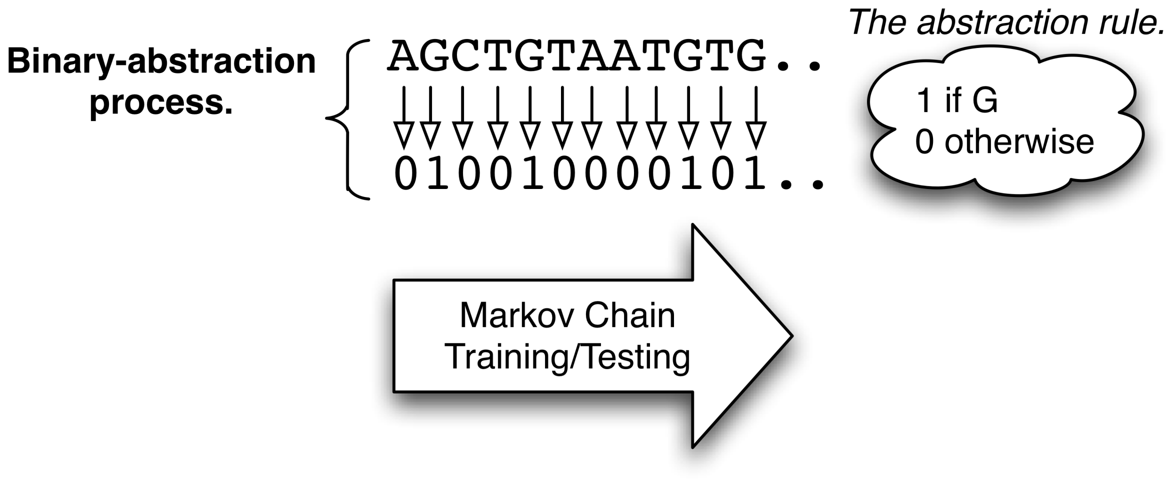
4.2.1 Binary-abstracted Markov models—BAMM

We have developed a method called "BAMM" (binary-abstracted Markov model) that can be used to analyze nucleotide sequences longer than traditional Markov models permit as well as selectively emphasize certain nucleotide patterns. This is accomplished by abstracting a selected portion of the nucleotide information into binary sequences that can be later used as input for the usual Markov chain algorithm. After the abstraction process, the nucleotide information being analyzed will effectively cover about 11 to 50 base pairs (depending on the given parameters) and will emphasize a specific portion of the nucleotide information according to the given abstraction rule. We discuss this process in greater detail along with our initial conceptions of a coding statistic algorithm [2].

4.2.1.1 The binary abstraction process

We define BAp to mean binary-abstraction of nucleotides on the level of p-mers. If we choose $p = 1$ (an "abstraction level" of 1 or "BA1" for short) then we will convert single nucleotides to 0 or 1 according to some *abstraction rule*. An explicit specification of an abstraction rule is called a "mapping" or "map" for short (see Table 4.11 for a $p = 2$ example). Abstraction rules need not be specified explicitly but can be written implicitly such as in the case of context-dependent abstraction rules (see Section 4.3.3) or the GT-rich abstraction rule ("check if the triplet is rich for G+T"; that is, "check if the number of G+T is ≥ 2"). Notwithstanding, implicit declarations of abstractions rules can always be reformulated to an explicit mapping (as we have done for maps based on triplet richness).

Figure 4-1: Nucleotide sequences are abstracted into binary ones using an abstraction rule. A Markov chain algorithm can then be trained and tested on the generated binary sequences for classification purposes. The abstraction rule pictured here (the G-map) is for an abstraction level of 1 nucleotide at a time.



A simple example of the abstraction or conversion process is given in Figure 4-

1. First the nucleotide sequence is converted into a binary one using the "G-map" abstraction rule, where all Gs are converted to 1 and all As, Cs, and Ts are converted to 0. The abstraction level is $p = 1$ since we convert only one nucleotide nucleotide at a time. After the binary sequence has been generated, it is used to train a Markov model and perform sequence classification.

Every abstraction rule contains two *conversion classes*, one for 1 and one for 0. According to the G-map example, $G \rightarrow 1$ belongs to the 1-conversion class while $A \rightarrow 0$ belongs to the 0-conversion class. Each unique abstraction of a nucleotide pattern to a bit is called a "conversion." Thus, the G-map abstraction rule contains 4 conversions (one for A, G, C, and T) and two conversion classes (1 and 0). The number of conversions for each abstraction rule is dependent on the abstraction level (4 for BA1, 16 for BA2, 64 for BA3, 256 for BA4, *et cetera*). The number of conversion classes remains constant at two classes but the actual cardinality (number of conversions per conversion class) depends on the particular abstraction rule. For example, using the G-map abstraction rule the cardinality of the 1-conversion class is 1 while the 0-conversion class cardinality is 3.

4.2.1.2 The original BAMM-like algorithm

Using the nucleotide-to-binary conversion process as previously described, we abstracted four binary sequence datasets from human genomic nucleotide sequences: *training* exon and intron datasets as well as *test* intron and exon datasets (see Table 4.2). The training intron and exon groups were used to produce frequency tables of the binary patterns in order to train the model (for a given abstraction rule). The test data was used to assess the model's accuracy. For a BA2 example, if we were interested in all two-bit patterns (11, 10, 01, and 00) we could scan along each converted binary sequence with a window of size 2 and advance bit by bit, counting the number of occurrences of each pattern. The number of occurrences per pattern could then be converted into frequency values. Thus, considering some frequency function f taking a binary pattern, $f(11) + f(10) + f(01) + f(00) = 1$. Given a sufficiently large training dataset, we can use the observed frequencies to estimate the probability of a binary pattern occurring in an exon $[P_E(\cdot)]$ versus an intron $[P_I(\cdot)]$.

Clearly, if $P_E(11) > P_I(11)$ then the pattern "11" is more likely to occur in an exon (otherwise not). If we instead use the log probability ratio, $\log \frac{P_E(11)}{P_I(11)}$ then a positive value suggests an exonic pattern while a negative value suggests that "11" is intronic. For similar values of P_E and P_I, the log odds ratio is close to zero. Consider some converted binary sequence "1101." If we are examining the sequence with a sliding window size of two, we have to analyze 11, 10, and 01. Using methods similar to those outlined in [2] we will calculate the average log probability ratio for the sequence 1101:

$$\begin{aligned}\text{Score}(1101) &= \frac{1}{3}\log \frac{P_E(11)\cdot P_E(10)\cdot P_E(01)}{P_I(11)\cdot P_I(10)\cdot P_E(01)} \\ &= \frac{1}{3}(\log \frac{P_E(11)}{P_I(11)} + \log \frac{P_E(10)}{P_I(10)} + \log \frac{P_E(01)}{P_E(01)})\end{aligned}$$

Let S be some binary sequence from the test set. For the purposes of sequence classification we considered any $\text{Score}(S) > 0$ to predict an exonic sequence while a $\text{Score}(S) \leq 0$ would predict an intron for the given test input sequence. Using *a priori* knowledge of which sequences are real exons and introns in the test set, we can see if the particular model/abstraction rule/window parameter is a very accurate choice for sequence classification. See Section 4.2.3.1 for more details on determining model/classifier goodness.

4.2.1.3 Markov chains and binary-abstracted sequences

In order to have a stronger mathematical basis (and indeed, slightly better accuracy) we modified our earlier BAMM-like algorithm to use Markov chain computations with the converted binary sequences. The use of a Markov model on the binary-abstracted sequences also allows us to directly compare our method with the more traditional homogeneous nucleotide Markov model (in terms of sequence classification comparisons, not sequence parsing).

Throughout this study Markov models of order k will be denoted as MMk. Thus, "MM5" would stand for a Markov model of order $k = 5$. Binary-abstracted Markov models have the additional p parameter for the abstraction level, so we will use the notational form BApMMk. Therefore, "BA3MM10" would have an abstraction level of $p = 3$ (nucleotide triplets) with a Markov model order of $k = 10$ (memory of 10 bits).

The Perl source code for the BAMM algorithm is given in the Appendix section 4.B.1, while our algorithm source code for a homogeneous nucleotide Markov model is given in Appendix section 4.B.2. Instead of going into great detail on the code, we will discuss the mathematical considerations for both algorithms here. The conditional probability of a particular input sequence S being exonic can be calculated using Bayes' rule. Equations 4.1 and 4.2 show the simplified solutions for $P(\text{exon}|\text{S})$ and $P(\text{intron}|\text{S})$. The prior probability of getting an exon or intron [$P(\text{exon})$ and $P(\text{intron})$ respectively] is taken to be the uniform value of $\frac{1}{2}$ since there are only two models being considered. While these prior probabilities do not have to be uniform, we have found this choice to work well.

$$\begin{aligned}
P(\text{exon}|S) &= \frac{P(S|\text{exon}) \cdot P(\text{exon})}{P(S|\text{exon}) \cdot P(\text{exon}) + P(S|\text{intron}) \cdot P(\text{intron})} \\
&= \frac{P(S|\text{exon}) \cdot \frac{1}{2}}{P(S|\text{exon}) \cdot \frac{1}{2} + P(S|\text{intron}) \cdot \frac{1}{2}} \\
&= \frac{P(S|\text{exon})}{P(S|\text{exon}) + P(S|\text{intron})} \qquad (4.1) \\
P(\text{intron}|S) &= \frac{P(S|\text{intron})}{P(S|\text{exon}) + P(S|\text{intron})} \qquad (4.2)
\end{aligned}$$

In order to calculate the score value for the binary sequence S, we can use the log-likelihood function by dividing Equations 4.1 and 4.2 and taking the log, similar to what we did in the previous section. Negative scores will be predicted to be intronic while positive scores will be predicted to be exonic. The natural log is here preferred because of some particulars in Perl programming efficieny. Equation 4.3 shows the simplification of the log-likelihood function. In order to compute the score value we must now turn to Markov chains.

$$\begin{aligned}
\text{Score}(S) &= \ln \frac{P(\text{model} = \text{exon}|\text{seq} = \text{S})}{P(\text{model} = \text{intron}|\text{seq} = \text{S})} \\
&= \ln \frac{P(S|\text{exon}) / P(S|\text{exon}) + P(S|\text{intron})}{P(S|\text{intron}) / P(S|\text{exon}) + P(S|\text{intron})} \\
&= \ln \frac{P(S|\text{exon})}{P(S|\text{intron})} \\
&= \ln P(S|\text{exon}) - \ln P(S|\text{intron}) \qquad (4.3)
\end{aligned}$$

Let us now consider a binary Markov chain of order $k = 1$ for some sequence S of length L. The notation S_1 would be the *first* bit in the sequence whereas S_L would be the *last* bit in the sequence. MM1 indicates that the model remembers one bit when it transitions to the next bit in the chain. Frequency values for both single bits and two-bit patterns must be calculated in order to create probability matrices for both the initial and transition probabilities. In order to compute Equation 4.3 we must compute two Markov chains for the given test sequence, one supposing it were an exon, and one supposing it were an intron. Equations 4.4 and 4.5 describe the simplified solution for these Markov chains, respectively. Consider again some binary sequence "1101," $P_E(S_1)$ would be the initial probability of 1 for the first bit in the sequence, given it were an exon. The transition probability $P_I(S_1 \rightarrow S_2)$ would describe the probability of going from the first bit, 1, to the second bit 1, given the sequence were an intron.

$$\begin{aligned}
\ln P(\text{seq}|\text{exon}) &= \ln\{P_E(S_1) \cdot P_E(S_2|S_1) \cdot P_E(S_3|S_2) \cdots P_E(S_L|S_{L-1})\} \\
&= \ln\{P_E(S_1) \cdot P_E(S_1 \rightarrow S_2) \cdots P_E(S_{L-1} \rightarrow S_L)\} \\
&= \ln P_E(S_1) + \ln P_E(S_1 \rightarrow S_2) + \cdots + \ln P_E(S_{L-1} \rightarrow S_L) \qquad (4.4) \\
\ln P(\text{seq}|\text{intron}) &= \ln P_I(S_1) + \sum_{b=1}^{L} \ln P_I(S_{b-1} \rightarrow S_b) \qquad (4.5)
\end{aligned}$$

Finally, suppose GGAG were the original nucleotide sequence and 1101 was the binary-abstracted sequence according to the G-map abstraction rule, then Equations 4.4 and 4.5 could be substituted into Equation 4.3 to calculate the score value for this BA1MM1 classifier. We compute this example as a help to the reader:

$$\begin{aligned}
\text{Score}(1101) &= \ln P(1101|\text{exon}) - \ln P(1101|\text{intron}) \\
&= (\ln P_E(1) + \ln P_E(1 \rightarrow 1) + \ln P_E(1 \rightarrow 0) + \ln P_E(0 \rightarrow 1)) \\
&\quad - (\ln P_I(1) + \ln P_I(1 \rightarrow 1) + \ln P_I(1 \rightarrow 0) + \ln P_I(0 \rightarrow 1))
\end{aligned}$$

4.2.2 Datasets and databases

We obtained exons, introns, and untranslated regions from our EID (exon-intron database) [14] and used the same intergenic dataset (human genome build 36) as in our study on genomic mid-range inhomogeneity [8]. We define "intergenic" regions as genomic regions occurring between genes. The source datasets were processed using three primary filters:

I. redundant intron and exon sequences were filtered out by recording the first and last 100 base pairs of each sequence (or the whole sequence if the total length was less than 100 bp) and then removing subsequently processed sequences that had duplicate tail or head sequence fragments;

II. modeled proteins (hypothetical ones) were removed when the "XP" RefSeq identifier was found in the header;

III. un-named proteins were filtered out when the "gene" attribute in the EID header contained "LOC..." for a locus identifier.

Table 4.1 shows the non-redundant, processed datasets for various human genomic regions. All of the data shown is from the direct or coding strand, although building Markov models to handle both strands in parallel is a useful advancement [7], we do not do so for the sake of simplicity. The CDS exon group refers to coding sequence exons while UTR ($5'$ or $3'$) exons refers to untranslated exons. Statistics are given for the total number of sequences, the total number of nucleotides, the average nucleotide length of the sequences for each genomic region and the standard deviation of the element lengths (SD). Introns that interrupt $5'$ untranslated regions are considered to reside within the $5'$ UTR; introns that interrupt $3'$ untranslated regions are considered to reside within the $3'$ UTR; and introns that interrupt the coding sequence are considered to reside within the CDS. Introns that interrupt the junction between the 5′ UTR and the CDS are considered to be within the CDS whereas introns that interrupt the junction between the CDS and the 3′ UTR are considered to be within the 3′ UTR.

Table 4.1: Non-redundant, processed datasets for various human genomic regions.

Dataset	#Sequences	Total Base Pairs	Average Size (bp)	SD
CDS Exons	168,893	26,966,425	159.7	233.0
5′ UTR Exons	24,473	3,317,321	135.6	170.6
3′ UTR Exons	19,456	16,137,666	829.4	1,002.0
All Introns	77,908	422,494,026	5,423.0	16,591.4
Introns in CDS	72,985	379,248,401	5,196.3	16,296.0
Introns in 5′ UTR	3,858	39,260,863	10,176.5	21,889.5
Introns in 3′ UTR	1,064	3,984,629	3,745.0	10,690.2
Intergenic Sample	1,114	35,159,606	31,561.6	62,586.4

Using the processed datasets from Table 4.1 we randomly selected training, validation, and test sets. Training data is used to build the model, validation data is used to optimize parameters such as abstraction rules, and test data is used to quantify the final measure of accuracy. Our selection algorithm requires a desired threshold for the total number of nucleotides (typically 12 Mb for training sets and 3 Mb for test & validation sets) and then randomly selects sequences from the dataset until that threshold is surpassed. A minimum length (per sequence) can also be specified. We originally used a minimum length of 39 bp and then later truncated the random datasets to a minimum length of 44 bp for full compatibility across multiple binary-abstraction methods (44 bp is the minimum sequence requirement for BA4MM10). The statistics for the random datasets is listed in Table 4.2. As before, we include the total number of sequences, the total length length of the dataset, and the average sequence size.

The random sampling process performs two additional, important operations: (1) sequences containing non-canonical bases, such as the ambiguous nucleotide ‘N’, are excluded; (2) coding exons are set to be in-frame or mixed frame depending on the desired dataset. This second point requires further elaboration. Introns are known

Table 4.2: Random datasets for model building and testing. All random datasets have mixed reading frame unless noted to be in-frame. Statistics are given for the total number of sequences, the total number of nucleotides for the set, and the average sequence length in base pairs.

Training sets: lengths $\geq$ 44 bp			
Dataset	# Sequences	Total Nucleotides	Average Size (bp)
Human CDS Exons	72,366[a]	11,965,969	165.4
Human Introns (any)	2,484	12,002,982	4832.1
Human 5′ UTRs Exons	7230	1,200,135	166.0
Human 5′ UTRs Introns	128	1,201,006	9,382.9
Test sets: lengths $\geq$ 44 bp			
Human CDS Exons	18,038	2,990,962	165.8
Human Introns (any)	586	3,003,840	5126.0
Human 5′ UTRs Exons	1819	300,037	164.9
Human 5′ UTRs Introns	62	302,955	4,886.4
Validation set (disjoint dataset used for optimization purposes).			
Human CDS Exons	17,773[a]	2,991,032	168.3
Human Introns (any)	752	3,000,827	3,990.5
Human in-frame training and test sets: lengths $\geq$ 39 bp			
Training CDS Exons	72,814	12,000,142	164.8
Training Introns (any)	2,273	12,001,894	5,280.2
Test CDS Exons	18,292	3,000,705	164.0
Test Introns(any)	543	3,002,327	5,529.1

[a] The same datasets with an extra 830 and 220 smaller exons respectively was also used for some optimizations.

to interrupt exons in three phases: in-frame (phase 0), after the first nucleotide in the codon (phase 1), and after the second nucleotide in the codon (phase 2). About 46% of our exons will be in-frame, around 32% will be interrupted at phase 1, and approximately 22% will be interrupted by phase 2 introns. This distribution of intron phases is about the same as that observed within human genome [15]. For the our "mixed frame" random datasets, we do not modify exons interrupted upstream by phase 0 introns (in-frame); we remove the first nucleotide from exons interrupted upstream by phase 1 introns, putting them in the third reading frame (frameshift 2); exons interrupted upstream by phase 2 introns have the first two nucleotides removed from the exon, putting them in the second reading frame (frameshift 1). Thus, a little over half of the coding exons will be out of frame (∼54%) and the rest will be in-frame. We also see that the exons interrupted upstream by phase 1 and 2 introns are modified such that the natural frameshift for the downstream un-spliced exon will be altered—in this way the exonic sequences in the dataset will not be correlated with their true upstream intron phase. However, the random chance of selecting an exon to be in-frame will still be the same as if we selected exons from a random pool of naturally occurring exons where the splice sites were known absolutely. For the in-frame dataset our algorithm merely cuts all exons interrupted by intron phase 1 and 2 to be in-frame.

4.2.3 Optimization of abstraction rules.

In the development of BAMM to do the discriminative scoring of exons and introns in eukaryotic species, we have leveraged the largest supercomputer in Ohio (the "Glenn" supercomputer with 4,212 Opteron CPUs) to run highly optimized C code for the brute force refinement of our discrimination models. Our work on optimizing 3-mer abstraction rules (including all trials) required a total 17 hours of supercomputer time on 512 logical cores (128 physical nodes). For the main run, we processed 33.5 million triplet abstraction rules (2^{25}) over 8 hours and 43 minutes. The abstraction rules were generated randomly within an interval spanning 2^{39} maps within the binary space of $\mathbb{B}^3 = \{0, 1\}^{64}$, which is $2^{64} \approx 1.8 \times 10^{19}$ elements large (see that ${}^{2^{64}}\!/_{2^{25}} = 2^{39}$). Recall that for triplets, $4^3 = 64$ nucleotide patterns will be converted to 0 or 1. Since each map has an inverted pair (the so-called "inverse map"), only half of the map space needs to be analyzed. This is easy to see for the trivial cases where all triplets are converted to 1's versus all triplets being converted to 0's—these abstraction rules will be equivalent in terms of their classification and, indeed, their uselessness.

The aforementioned BA3 optimization is known as a "random search" and is useful for large spaces where a sufficient number of "good" solutions may be thought to exist [16]. The C-code actually implemented in the random search was for a BAMM-like precursor to our BAMM algorithm. The details are listed in Section 4.2.1.2. The results from our random search produced a number of "good" abstraction rules (see Section 4.2.3.1). These were used as seeds for a binary particle swarm optimization algorithm in Section 4.2.3.2.

4.2.3.1 Measuring model goodness.

For all of our model optimizations we have utilized a unique measure of "goodness" called the M-value (or just M for short) as listed in Equation 4.6 based on both sensitivity and specificity values. Sensitivity (SN) is defined as the $SN = \frac{TP}{TP+FN}$ where TP is the number of true positives (correct predictions for the class of interest, *e.g.*, coding exons) and FN is the number of false negatives (incorrectly assigned predictions for the class of interest).

Similarly, specificity SP may be defined as $SP = \frac{TN}{TN+FP}$ where TN would be the number of true negatives and FP would be the number of false positives. It should be noted that the SP is also defined using the formula for the positive predictive value (PPV) within the literature [13], which is a choice natural considering the emphasis of finding coding sequences in gene finding. Using the PPV, specificity is defined by the ratio of true positives to all possible positive results: $SP = \frac{TP}{TP+FP}$.

It is obvious that for a classifier or prediction model to be accurate, one wants to maximize both the sensitivity and specificity. A general measure of accuracy [17] ($\frac{TP+TN}{TP+TN+FP+FN}$) could not be used since the number of exons will greatly exceed the number introns, although the nucleotide sample used is the about the same for both groups (see Table 4.2). One can still measure accuracy using various method such as correlation coefficients, averages, and ACP values. However, we elected to design our own overall measure of accuracy (the "M-value") based upon our optimization goals and needs (see Equation 4.6). First, M should be in the unit interval of $[0, 1]$ with

SN	SP	Average(SP, SN)	M-value
0.50	0	0.25	0.21
0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50
0.50	0.75	0.63	0.60
0.50	0.85	0.68	0.63
0.50	1.00	0.75	0.65

Table 4.3: Averaging the SN and SP versus taking the M-value. Percentages are displayed as decimals. The M-value is not a percentage.

larger values meaning better prediction accuracy. Values of M approaching 0 indicate a mis-labeling of the classes or anti-prediction. Therefore, a value approaching 0.50 is the most random.

Second, we wanted a measure of goodness that combined both SN and SP in a way that emphasizing one over the other would reduce the goodness; that is to say, we wanted a balanced way of measuring overall model accuracy. An average of SN and SP is an effective, simple way to measure classifier goodness. We show the behavior of the M-value (equation below) versus the average of SN and SP in Table 4.3 for constant SN and varying SP. Observe that when $SP = 0.50$ and $SN = 1.00$ the average is 0.75 while M is 0.65, a much more conservative value given the extreme imbalance in SN and SP. If SN and SP were both in balance, say, at 0.75, then both the average and M-value would be at 0.75.

$$SN = \text{exon accuracy},\ SP = \text{intron accuracy}$$

$$\begin{aligned} M\text{-value} = &\ 1 - \frac{\sqrt{(SN-1)^2 + (SP-1)^2}}{\sqrt{2}} \\ = &\ 1 - \sqrt{\frac{(SN-1)^2 + (SP-1)^2}{2}} \end{aligned} \tag{4.6}$$

Equation (4.6) shows the M-value. Exon accuracy (the number of exons correctly predicted out of all exons) is taken to be the sensitivity while intron accuracy (the number of introns correctly predicted out of all introns) is equivalent to the specificity. The best possible accuracy of any classifier will be at point $(1,1)$ when you view specificity values (y-axis) versus sensitivity (x-axis) on a two-dimensional plot. An ROC curve is a kind of graph for $(1-SP)$ versus SN that is commonly utilized [17,18] to characterize the optimal value of (SN, SP) pairs by evaluating the area under the curve. Similarly, our M-value uses a geometric approach by first taking the euclidean distance of an (SN, SP) pair from the best possible pair $(1,1)$, normalizing it to the unit interval $[0,1]$, and complementing it to make larger values "better." We have observed an almost identical classifier assessment technique as ours within the literature [19]. The M-value function will, of course, be symmetric.

4.2.3.2 Finding BA3 abstraction rules with binary particle swarm optimization.

In order to find the BA3 best abstraction rule and to seed round 1 of the BA4 optimization trial 1, we implemented a binary particle swarm optimization (BPSO) program in the C programming language [20]. The general particle swarm optimization algorithm (PSO) relies on multiple particles in a swarm "following" the currently best particle in order to get better and better results (like bees in a swarm). Each particle may contain multiple "dimensions" of real numbers that determine the particle's velocity. Convergence toward the best particle is limited by a maximum velocity parameter. Particle swarm optimization has several advantages over other techniques such genetic algorithms: it has fewer parameters to adjust, more diversity in the population (to avoid local optima), and each particle remember its local best in addition to the globally best solution [21].

Unlike the normal PSO algorithm, the particle dimensions for the BPSO algorithm are binary numbers (1 or 0) instead of real numbers, and so the changing of any dimension (to be more like the best particle) means flipping a bit instead of changing the value of a scalar. The changing of the dimension is accomplished by changing the *probability* that a bit will flip. Thus, the more certain we are that a bit value (for a particular dimension) is good, the higher the probability will be that all particles will change to that bit value to follow suit (making good values become fixed over time). For the BA3 abstraction rule optimization, each BPSO particle was actually an abstraction rule represented as a binary string. Each bit in the string corresponded to a particular conversion (say, GGG $\rightarrow$ 1). Considering a binary abstraction level of 3, each binary string will be 64 bits or 64 binary particle dimensions. In order to find the BA3 best map we used 16 random seeds (particles) with a maximum velocity of 6 (among other trial parameters). The optimal value was obtained after 295 iterations (maximum 500 iterations). Using seeds obtained from our random BA3 search as well as using a bit-mutating version of the BPSO did not produce better results [22]. Moreover, the BPSO was a much more computationally inexpensive way to optimize the BA3 abstraction rules than the use a random search. For a brief comparison, the best results of the random search yielded an M-value of 0.787 (under the validation set and our initial BAMM-like algorithm) while the BPSO algorithm was able to obtain an M-value of 0.829 using the same dataset and algorithm.

We also attempted to use the same BPSO technique to optimize BA4 abstraction rules but the number of dimensions (256) was, we believe, prohibitively large for that optimization to come close to any kind of global convergence. For a description of our alternative strategy, please see Section 4.2.3.4.

4.2.3.3 BA2 Optimization

In order to find the BA2 best abstraction rule we exhaustively searched the abstraction rule space using the Glenn supercomputer. The process took 7 hours and 16 minutes by supercomputer for what would have taken over 38 days otherwise. We used the current BAMM algorithm to test all 32,768 abstraction rules on the validation dataset and sorted the results by M-value. The best result (BA2 best) is displayed in Table 4.15. The reader will notice that $2^{16} = 65,536$, not 32,768. This is again because we exclude all the inversions within the abstraction rule space, since they produce equivalent classifiers.

4.2.3.4 BA4 Optimization

For DNA tetranucleotide abstraction rule optimization (BA4), computational requirements were even greater than for the BA3 random search. Thus, a new hill-climbing strategy was developed that took the best map of the previous round as a "seed map" and then generated all maps within a Hamming distance of 1 (256), 2 (32,640) and 3 (2,763,520) for processing in the current round. (See Section 4.3.1 for a discussion of Hamming distance in optimization.) The Glenn supercomputer therefore analyzed 2,763,520 maps per round. We chose to perform three different optimization trials starting from three initial seed values. Trial 1 was obtained from our failed attempts at binary particle swarm optimization (basically, a random map). Using the ordering of tetranucleotide conversions as seen in the source code of Algorithm 4.B.1 (observe that *enumerate* sorts each 4-mer A before T before C before G), the trial 2 seed map was chosen in a arbitrary manner by assigning the first half of the conversions to the 1-conversion class and the rest to the 0-conversion class. We called this seed the "half & half" seed. Similarly, the initial seed for the BA4 optimization trial 3 was generated by choosing 1 for the first tetranucleotide conversion, 0 for the next, 1 for the following, and so forth. We refer to this seed as the "striped" seed in the tabular data. While choosing an arbitrary or random initial seed value is not necessary, our choosing them does show the general requirements for convergence.

The trials required approximately 116 individual supercomputer jobs each using 128 computer cores (32 physical nodes) and taking a little over 2 hours of wall time per round. The total wall time for all tetranucleotide abstraction rule optimization took over 10 and a half days for 324 million abstraction rules. This was done in 33, 46, and 41 rounds for trials 1, 2 and 3 respectively. The results are presented in Section 4.3.2.

4.2.4 An *a priori* method for constructing abstraction rules

In order to generate non-random abstraction rules both with biological meaning as well as decent accuracy, we define the *a priori* abstraction rule methodology, as suggested by our colleagues at GA Tech (the Mark Borodovsky lab). The first step is to analyze the n-mer frequency distribution in both of the sequence groupings. For us, this will be coding exons and introns, but this need not be the case. Suppose we want to build an *a priori* map for the binary-abstraction level of 1 (BA1), we can therefore analyze the abundances of A, T, C, and G in both the coding exon and intron groups. The process to generate the *a priori* 1 abstraction rule is shown Table 4.4 with frequencies taken from the whole CDS exon and intron datasets as listed in Table 4.1. If the frequency of the DNA pattern (in this case a single nucleotide) is greater in the intron group, then the conversion is set to 0, otherwise if the exon group is greater or equal in frequency, then the conversion will be set to 1.

Table 4.4: How to generate the *a priori* 1 abstraction rule.

Base	Exon Freq.		Intron Freq.		Conversion
A	26.1%	$<$	27.8%	$\Rightarrow$	0
T	21.9%	$<$	30.8%	$\Rightarrow$	0
C	25.7%	$\geq$	20.2%	$\Rightarrow$	1
G	26.3%	$\geq$	21.2%	$\Rightarrow$	1

The *a priori* 1 abstraction rule, or AP1 for short, is the CG-mapping rule for BA1. Choosing G+C content is a natural and biologically meaningful choice for an abstraction rule. However, as we will later see in Table 4.10, the CG-map is not the best abstraction rule for BA1. This is probably due to the fact that we randomly choose exons and introns for our training and testing set from across the entire human genome, which means that we will span multiple GC-isochores within our datasets.

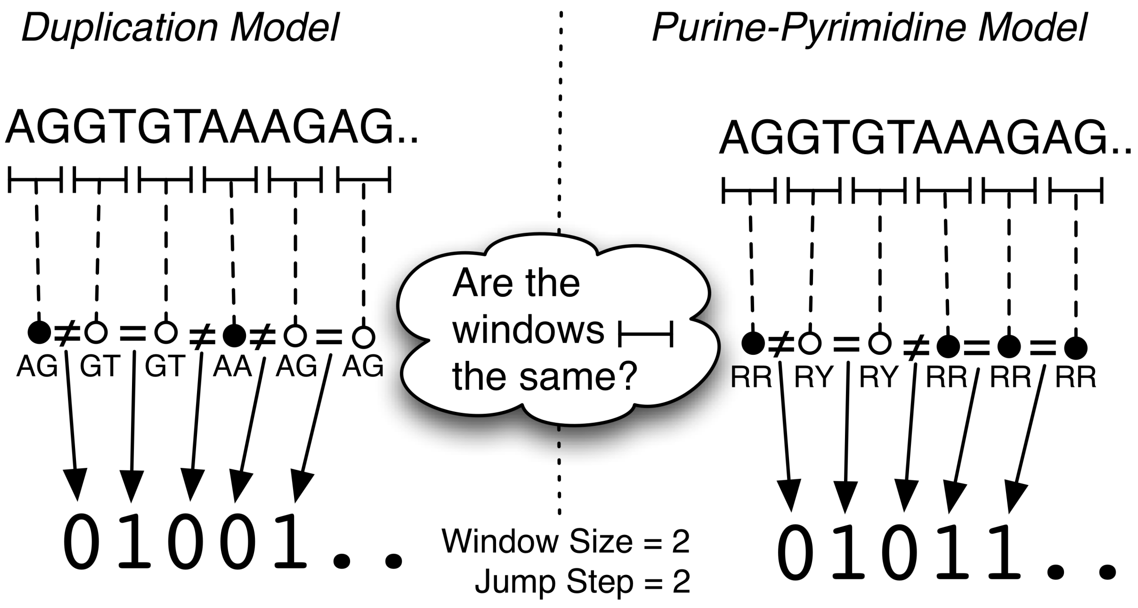
4.2.5 Context-dependent methods for BAMM

Context-dependent binary-abstracted Markov models (CDBAMM) specify abstraction rules based on the context of two adjacent nucleotide windows. In other words, conversion to 0 or 1 is based on whether the two adjacent nucleotide windows are the same/similar in terms of their nucleotide sequences or if they are different. The two major parameters are window size and jump step. The *window size* is usually 1 to 4 bases and governs how much information will be compared at a time. The *jump step* parameter determines where the next "adjacent" window will occur. For example, suppose we have the sequence "AGGT," with a window size of 2 and a jump step of 1. The first window will cover AG while the second window (jumping only 1 nucleotide) will cover GG. If the jump step had been 2, the second window would have covered GT. Whenever the jump step equals the window size the windows will be non-overlapping (similar to traditional BAMM) while a jump step less than the window size means that there will be some overlapping bases being compared. Each particular window size/jump step combination can capture a particular biological event or sequence property. For example, a window size of 1 and jump step of 1 will emphasize the occurrences of single homonucleotide repeats or duplication events.

We describe two basic CDBAMM abstraction rules or models: the duplication abstraction rule (testing for window equality), and the purine-pyrimidine abstraction rule (testing if the windows are similar; that is, in terms of purines and pyrimidines). Figure 4-2 shows the conversion process for these two models.

In the left panel, Figure 4-2 shows the duplication abstraction rule for a window size of 2 and a jump step of 2 where adjacent, non-overlapping dinucleotides are compared for equality or inequality. For the first two windows, observe that AG $\neq$ GT so a 0 is generated for inequality in the two adjacent windows. The algorithm

Figure 4-2: The abstraction process of context-dependent binary-abstracted Markov models. The left panel shows the duplication model while the right side shows the purine-pyrimidine abstraction rule. Here, the window size is 2 nucleotides; the jump step to the next window is also 2 base pairs.



now jumps two bases such that the second window (GT) is compared to the third window (GT) and generates a 1 for window equality. This process continues until the entire sequence has been analyzed.

The purine-pyrimidine abstraction rule (Figure 4-2, right panel) works the in exactly the same manner as the duplication model except that nucleotides are first generalized into purine (R) or pyrimidine (Y) before the windows are compared. Thus, exact sequence repetition is not analyzed, only a *similarity* in repetition (purine-pyrimidine). The difference between the two models can be observed in the fourth and fifth windows ("AAAG") where for the duplication model has AA $\neq$ AG and generates a 0. In the purine-pyrimidine model the generalization process will produce a match of (AA =) RR = RR (= AG) and generate a 1 instead. After the binary sequence is generated the Markov chain algorithm can be employed in the same manner as the traditional BAMM algorithm to do the actual model training and sequence classification.

4.2.5.1 Empty Probabilities ($P_\emptyset$)

Unfortunately for certain choices of parameters and abstraction rules, CDBAMM (as well as BAMM) will start experiencing a significant number of *empty probabilities* (denoted $P_\emptyset$) when evaluating the test set. An empty probability is an untrained variable [such as $P_E(S_i \rightarrow S_{i+1})$ from Equation 4.4] within the Markov chain. Having a large enough training set will usually ensure none of these $P_\emptyset$ will occur, but that is not always the case. There are two easy ways of dealing with empty probabilities other than by lowering the Markov model order: ignore any sequence in the test set that contains empty probabilities, or, ignore the $P_\emptyset$ themselves within the Markov chain computation. We prefer the latter method for CDBAMM and BAMM, although selecting a sufficiently small Markov model order (usually $k < 14$ for BAMM and $k < 7$ for nucleotide Markov models) does minimize the number of $P_\emptyset$. Unfortunately, even with a smaller Markov model order and a large training set (12 Mb per group), empty probabilities may still occur because of the nature of the specified abstraction rule or CDBAMM parameters. At this point, ignoring the $P_\emptyset$ may cause some mild to severe inaccuracy in the model although the role of $P_\emptyset$ in the model accuracy may be hard to tease out. We do know that model accuracy is eventually lowered when one increasingly chooses inappropriately large Markov model orders for BAMM or for a homogeneous nucleotide Markov models.

In the case CDBAMM, we used BAMM4 for the DUP model whereas we used BAMM6 for the YR abstraction rule. These are modest orders considering our 12 megabases of training data. However, due to the nature of these CDBAMM abstraction rules, at certain parameters empty probabilities were quite abundant. We show in Table 4.5 the empty probabilities (their total number of occurrences, not the number of untrained variables) that complement the model accuracy results later listed in Table 4.16. In the table, the "Window" stands for the window size and the "Jump" stands for the jump step. The main point of the table is to show that the choice of parameters greatly impacts the number of empty probabilities (and accordingly the kinds of DNA patterns being abstracted), even for CDBAMM models with modest Markov model orders (4 & 6).

Table 4.5: The number of empty probabilities for the CDBAMM duplication abstraction rule (DUP) and purine-pyrimidine model (YR). The number for the empty probabilities ($P_\emptyset$) are for the total number of *occurrences* of untrained variables encountered while evaluating the test set. This is different from the total number of untrained variables. This table complements the results shown in Table 4.16

CDBAMM Parameters	DUP $P_\emptyset$	YR $P_\emptyset$
Window 1 Jump 1	0	0
Window 2 Jump 1	700,084	1,547,422
Window 2 Jump 2	0	0
Window 3 Jump 1	448,854	1,810,846
Window 3 Jump 2	0	0
Window 3 Jump 3	0	0
Window 4 Jump 1	217,078	1,542,806
Window 4 Jump 2	36,904	307,760
Window 4 Jump 3	0	0
Window 4 Jump 4	3,921	45

4.2.6 Support vector machines and model combination

We now outline the methodology used to combine classifier models under support vector machine (SVM) technology. The SVM was implemented under the free, large-scale machine-learning toolkit known as the SHOGUN project [23]. Specifically, we ran the SHOGUN interface written for Octave (a free open-source version of Matlab, see `http://www.octave.org`). As was recommended in the literature [24], we first employed a grid search of the parameters on the gaussian and then sigmoid kernels before arriving at the polynomial kernel as the best solution for our data domain. The final set of parameters of the SVM was to use the non-homogeneous polynomial kernel of degree 3 with normalization turned on.

The SVM was trained and tested with score values only and not with primary sequence data (such as the binary or nucleotide sequences themselves). The consequence of this is that although the dataset groups (exons and introns) are very similar in terms of their number of nucleotides, they are very different in terms of their number of sequences. If one trains with the exon and intron scores as is, the SVM will be biased to detect exons and so accuracy will be lost. The literature refers to this situation as an "unbalanced dataset" and lists it is a difficult problem to handle [25]. Of the solutions mentioned we chose to *down-sample* or reduce the number of exons to match the number of introns.

For the training phase of the SVM we used the usual training dataset to train each classifier (such as the BA3MM10 best map) and then used the same classifier to test on the training data itself (this is indeed the only fair way to do it if one only has *a priori* knowledge of the identities of the training sequences, as would be the case in a real world situation). The SVM then uses the score values (log-likelihoods) of these sequences to train itself. In terms of machine-learning terminology, each sequence in the self-tested training set will produce a data point with a field (score value) that corresponds to the prediction of each classifier. Therefore, the dimensionality of the data is the number of classifiers used, the number of data points is with respect to the number of sequences classified. Specifically, all 2,484 introns were used along with the first 2,484 exons (down-sampling) found in the training set. Since the training set is a random dataset, the selection of the first sequences should not bias the results in any way. This 1:1 ratio for the down-sampling produced the most balanced SVM training in our experience.

After the SVM was trained a similar procedure was applied to classify each sequence. First, the same classifiers as previously applied to the training of the SVM were applied to score each sequence in the test or validation sets (depending on which was used). No down-sampling is needed for the test and validation sets since the SVM is already trained. Next, if two classifiers (say, BA3MM10 and AP3) produced scores of +1 and +2 respectively for some particular exon in the test or validation set, then the SVM would make its own prediction based on the score value tuple (+1, +2) for that exon. If the SVM result was positive, the prediction would be counted as exonic while a negative score would predict for an intron. This process continued until all sequences had been classified as positive or negative according to the SVM. The usual M-value and exon/intron accuracy could then be estimated for the chosen set of classifiers. In short, the SVM both trains and tests sequence data using the predictions of other classifiers—such as BAMM and the homogeneous nucleotide Markov model—although it is not limited to classification based on these score values alone. Other data can be added as well into the final prediction such as sequence length, splice site scores, *et cetera*. This sort of SVM methodology (on different types of biological sequence signals) is used in full-fledged gene finding systems like mGene [12].

Finally, for Table 4.18 we performed a K classifiers out of N analysis $[\binom{N}{K}]$ to see which combination was the best. The selection of the best model combination was out of the total number of $\binom{N}{K}$ classifiers for each fixed K. We started with the $N = 10$ classifiers as listed in Table 4.17. Using a custom Octave script we tested every combination for $K = 1$ or 9 (10 combinations), $K = 2$ or 8 (45 combinations), $K = 3$ or 7 (120), $K = 4$ or 6 (210), $K = 5$ (252), and the single test of all 10 classifiers.

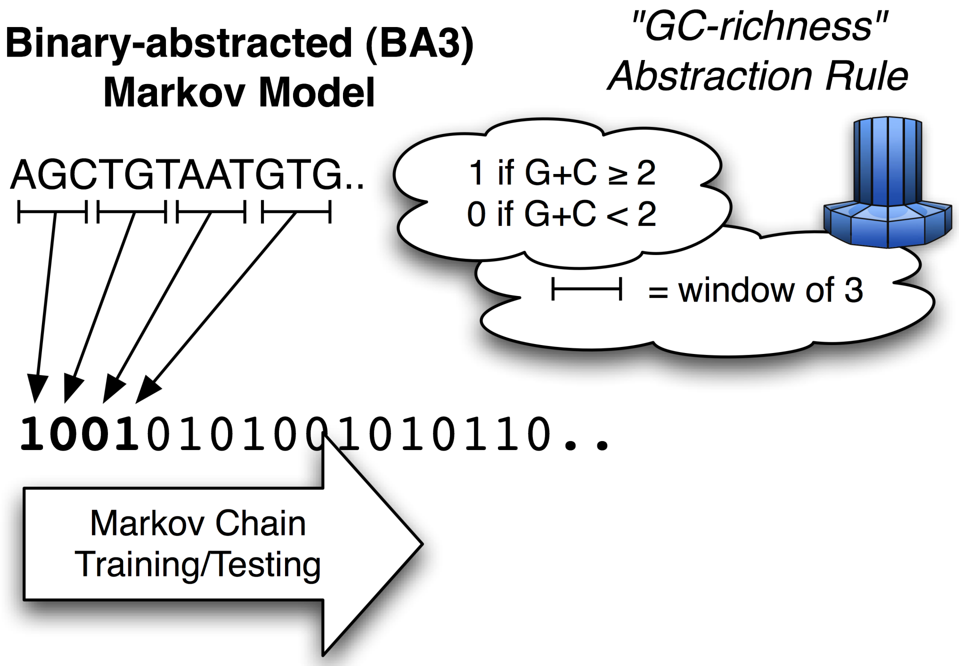
4.3 Results & Discussion

Given the nature of the genetic code, it is natural for us to first consider triplets. However, we shall do so in a frame insensitive manner. It is well-understood that long open reading frames and three base pair periodicity are indicators of coding sequences [26]. In order to complement what has been done, the majority of our results, though not all, will be devoted to classifying mixed frame coding exons versus introns (all human). We will also explore the efficacy of combining our classifiers under machine-learning techniques such as support vector machines as well as test our classifiers on untranslated regions.

4.3.1 Binary-abstraction of triplets

A common feature of higher eukaryotic genomes is long-range sequence patterns known as isochores (300+ kb regions with stable G+C composition [27]). Furthermore, G+C content clusters into G+C poor or rich regions at mid-range scales of 30 to 10,000 bp [8]. This being the case, one could devise an abstraction rule that emphasizes G+C richness using our binary conversion process on the level triplets. Figure 4-3 diagrams this abstraction rule. First each non-overlapping window (triplet, not necessarily in frame) is analyzed such that if it contains 2 or more G or Cs, it is converted to 1 otherwise 0. The usual Markov chain algorithm can then be applied to the generated binary sequences.

Figure 4-3: A diagram of the GC-rich abstraction rule. Triplets that are G+C-rich (≥ 2) are converted to 1, otherwise 0. The usual Markov chain algorithm can then be trained and tested on the generated binary sequences for classification purposes.


Binary-abstracted (BA3) Markov Model
"GC-richness" Abstraction Rule
AGCTGTAATGTG..
1 if G+C ≥ 2
0 if G+C < 2
= window of 3
1001010100101 0110..
Markov Chain Training/Testing

Since each bit value of 1 would denote a GC-rich triplet, then by examining just 10 bits (30 nucleotides) we would be able to capture middle-range GC-clustering in addition to shorter range GC-rich patterns. For any binary-abstracted Markov model of order 10 at the triplet level, there would be $2^{11} = 2,048$ possible bit patterns and at least as many transition variables for the algorithm to train (the initial probabilities require less sequence information, so they are not discussed). Compare this to the usual homogeneous nucleotide Markov model of order 6 (MM6) that analyzes $4^7 = 2^{14} = 16,384$ possible nucleotide patterns and has as many transition variables being needed to train. It would take choosing a higher order Markov model for the GC-rich map, say BA3MM13 (2^{14} bit patterns), to be roughly equivalent to MM6 in terms of the number of transition variables that need to be trained—albeit different information will be analyzed by both algorithms. Next consider the quotient of information obtained from a 75 bp training sequence. For a 75 bp nucleotide sequence, triplet abstraction rules will generate 25 bits ($\frac{75\text{ bp}}{3\text{ bp per bit}}$). A BA3MM10 classifier will analyze 11 bits at a time, meaning that a 0.0068 quotient of the transition variables will be trained ($\frac{25\text{ bits} - 11\text{ bit window}}{2^{11}\text{ t. variables}} = \frac{14\text{ observations}}{2048\text{ t. variables}}$) while for an MM6 classifier using the same 75 bp sequence, a 0.0042 quotient of the transition variables will be trained ($\frac{75\text{ bp} - 7\text{ bp window}}{4^{7}\text{ t. variables}} = \frac{68\text{ observations}}{16384\text{ t. variables}}$). Therefore, we have done *more* training of our Markov chain variables (proportionately) under BA3MM10 than with the homogeneous nucleotide Markov model of order 6, even with fewer actual sequence observations to use. On the other hand, the minimum input sequence length of BA3MM10 must be no less than 30 bp whereas MM6 requires no less than 6 bp for some sort of evaluation or training.

In any case, we observe that it is easy to use the biological property of G+C-richness (Figure 4-3) to create an abstraction rule for triplets. We have similarly devised abstractions rules based on G+T-richness as well as A+G-richness. The accuracies of these “richness” abstraction rules are presented in Table 4.6 for coding exons versus introns. Again, the accuracy values listed are not for a sequence parse but merely the accuracy of correct predictions for each group given the individual exon and intron sequences. The M-value in the last column of Table 4.6 represents the general accuracy of each model/abstraction rule (sort of like a special average for intron and exon accuracy) and is explained in detail in Section 4.2.3.1.

Another biologically interesting feature of nucleotide sequences are exonic and intronic splicing silencers and enhancers. Previous work has established a database of mutations that affect such splicing by observing experimentally validated exon retention/skipping events triggered by mutation (The Alternative Splicing Mutation Database, [28]). A “splicing potential” (SP) score is given to triplet motifs based on a triplet’s potential to affect the inclusion of exons [29] under alternative splicing. Splicing potential scores generated with negative scores are associated with mutations involved in exon skipping while positives scores are associated with exon inclusion. To create another abstraction rule, we took the 24 most positively scoring triplets from an updated version (in 2009) of our database [29] and converted those triplets to 1 else 0. We refer to the model generated by this abstraction rule to be the “Splicing Potential Top 24 Positive” map or just “Pos” for short. Its accuracy is also included in Table 4.6.

The best abstraction rule for the triplet abstraction level was found using optimization techniques such as the binary particle swarm optimization algorithm as described in Section 4.2.3.2. We refer to this mapping (explicit defined abstraction rule) as the “BA3 best” map or just BA3 for short. Finally, we analyzed the short-range triplet abundances in both introns and coding exons to create an *a priori* abstraction rule, assigning 1 to triplets more abundant in exons and 0 to triplets more abundant in introns (see Section 4.2.4). We refer to this abstraction rule or mapping as “*a priori* 3” or “AP3” for short. The BA3 and AP3 abstraction rule accuracies finish out Table 4.6.

Table 4.6: The accuracy of various BA3MM10 (abstraction of triplets, Markov order 10) abstraction rules on coding exons versus introns. Accuracies for introns and coding exons are displayed as a percentage of the true predictions in each group while the M-value represents a balanced measure of the classifier's overall accuracy.

Abstraction Rule / Model	Exon Accuracy	Intron Accuracy	M-value
GC-richness	68%	65%	0.665
GT-richness	65	83	0.725
AG-richness	70	71	0.707
SP Top 24 Positive	73	86	0.782
BA3 Best	77	93	0.831
A priori 3	76	69	0.726

It is clear from Table 4.6 that, along with the *a priori* 3 abstraction rule, only mediocre accuracy can be achieved with abstraction rules modeling nucleotide richness (M-values < 0.75). This may in part be due to the random sampling of the training set dipping into multiple GC-isochores (see Table 4.2). Interestingly though, the GT-richness abstraction rule has much better intron accuracy (83%) implying GT-clustering patterns in introns may be usable for sequence classification. Indeed, exonic splicing silencers are GT-rich [29]. The BA3 best and splicing potential maps perform much better ($M > 0.75$) with impressive accuracy in the BA3 best map for introns (93%).

In order to explore the splicing signals observed in [29], we created and tested a number of splicing potential abstraction rules. For the abstraction rule called "splicing potential (SP) version 2008," we used all triplets with positive splicing potential scores for the 1-conversion class (mapped to 1), otherwise 0—as reported in Table 1 of [29] at the time of publishing. We optimized this abstraction rule up to a Hamming distance (HD) of 4, taking the best result from flipping all combinations of 1, 2, 3, or 4 conversions at a time within the map (such as CCC $\rightarrow$ 0 becoming CCC $\rightarrow$ 1 for a single example of the Hamming distance 1 case). A "Hamming distance" is a computer science term used to mean the number of differences between two strings of bits. For example, for the bits $A = 010$ and $B = 110$, the Hamming distance between A and B is 1 because they differ in only the first bit. The optimized abstraction rule will therefore be no more than 4 conversions (bits) different than the original splicing potential abstraction rule on which it is based. After updating the Alternative Splicing Mutation Database in 2009, the new splicing potential scores were used to make an abstraction rule entitled "SP version 2009" in the same manner as the original SP version 2008 abstraction rule. SP version '09 was also optimized to a Hamming distance of 4.

Table 4.7: The accuracy of various splicing potential (SP) models as well as their optimized versions (to a Hamming distance of 4).

Abstraction Rule / Model	Exon Accuracy	Intron Accuracy	M-value
SP version 2008	64%	79%	0.704
SP '08 Optimized (HD4)	71	82	0.760
SP version 2009	72	85	0.777
SP '09 Optimized (HD4)	75	90	0.810
SP '09 Top 24 Positive	73	86	0.782
SP '09 Positive HD4	75	90	0.811
SP '09 Top 24 Negative	77	78	0.772
SP '09 Negative HD4	78	88	0.820
BA3 Best	*77*	*93*	*0.831*

From the SP '09 abstraction rule we also decided to emphasize negative and positive scores by creating abstraction rules emphasizing the top 24 most positive or top 24 most negative splicing potential scores. These are listed as the "SP '09 Top 24 Positive" and "SP '09 Top 24 Negative" abstraction rules respectively. Both of these maps were optimized to a Hamming distance of 4 as well. The accuracies of all of these splicing potential abstraction rules, along with the BA3 best map (for comparison), is given in Table 4.7. We notice that none of the SP abstraction rules are as good as our optimized BA3 best abstraction rule. Moreover, optimizing every splicing potential map produced a noticeable increase in accuracy, such as the "SP '09 Top 24 Negative" map going from an M-value of 0.772 to an M of 0.820, which had the highest score of all splicing potential abstraction rules. Gratifyingly, the update of our Alternative Splicing Mutation Database from 2008 to 2009 produced a better abstraction rule for classification purposes (M of 0.704 for SP '08 versus M of 0.777 for SP '09). This implies that the studied splicing signals were strong enough in terms of BAMM classification to have a direct impact on discriminating coding exons versus introns.

A natural question is how different these abstraction rules actually are. The Hamming distance matrix in Table 4.8 answers this question, detailing the number of differences in the splicing potential abstraction rules that convert nucleotide sequences to binary ones (plus the BA3 best map for comparison). The Hamming distance is symmetric, meaning that $\delta(A, B) = \delta(B, A)$ given some abstraction rules A and B, and it also contains the properties that $\delta(A, B) = 0 \iff A = B$ and $\delta(A, B) > 0 \iff A \neq B$. The maximum Hamming distance possible between our abstraction rules will depend on the number of unique nucleotide sequences to be converted, for triplets, that means 64, although considering inversions, 32 would be the most distant.

Table 4.8: The Hamming distance matrix for various splicing potential (SP) maps. "Neg" stands for the Top 24 Negative SP scores; "Pos" stands for Top 24 Positive, "Opt" stands for Optimized versions of the map (Hamming distance of 4); and the 08 and 09 abbreviations refer to the different releases of the Alternative Splicing Mutation Database.

$\delta(M_i, M_j)$	SP08	Opt08	SP09	Opt09	Pos	PosOpt	Neg	NegOpt	*BA3*
SP08	0	4	11	15	13	15	15	17	20
Opt08	4	0	9	11	11	11	13	15	18
SP09	11	9	0	4	6	10	10	12	19
Opt09	15	11	4	0	10	10	8	10	17
Pos	13	11	6	10	0	4	16	16	23
PosOpt	15	11	10	10	4	0	14	14	19
Neg	15	13	10	8	16	14	0	4	11
NegOpt	17	15	12	10	16	14	4	0	9
BA3	20	18	19	17	23	19	11	9	0

From Table 4.8 we see that all optimized versions of the SP maps are indeed 4 conversions (HD4) different than their original counterparts. More interestingly, as each SP map is optimized it gets closer and closer to the BA3 best abstraction rule. The "SP '09 Top 24 Negative" abstraction rule (called "Neg" in the table) goes from being only 11 bits different to being 9 different after being optimized versus the BA3 best abstraction rule; the "Pos" map goes from a Hamming distance of 23 to 19 versus the BA3 best map; the original SP '09 map (not emphasizing negative or positive splicing potential scores) goes from a Hamming distance of 19 to 17 versus the BA3 best map after optimization; and even SP '08 goes from an HD of 20 to 18. These results imply that our BA3 best abstraction rule exploits some splicing signals (perhaps silencer ones given its efficacy within introns) to do sequence classification. Moreover, better splicing potential maps are, in general, closer to the BA3 best map, although this is not always the case (SP '08 has an M-value of 0.704 and a Hamming distance of 20 from the BA3 best map while the "Pos" map has a Hamming distance of 23 and an M-value of 0.782). This suggests certain conversions or DNA triplets may be of more importance than others to the sequence classification process, but which?

Table 4.9 answers this question by outlining explicitly the abstraction rules from Table 4.7. These explicit definitions of the abstraction rules are also known as the mappings or maps. The 2nd column and on gives the binary conversion of the triplets listed in the first column for the particular splicing potential abstraction rule. The optimized versions of each map are listed side-by-side (to the right) for easy comparison with an arrow showing the changed or optimized conversion in the abstraction rule.

Table 4.9: Abstraction rules (triplets) for the splicing potential (SP) models of binary-abstracted Markov models. Arrows indicated the bits that changed due to the optimization (Opt) process.

Triplet	SP08	Opt08	SP09	Opt09	Pos	PosOpt	Neg	NegOpt	*BA3*
AAA	0	0	0	0	0	0	0	$\rightarrow 1$	1
AAC	1	1	1	1	0	0	1	1	1
AAG	1	1	1	1	1	1	1	1	1
AAT	1	1	0	0	0	0	0	0	1

Continued on next page

Table 4.9: (continued)

Triplet	SP08	Opt08	SP09	Opt09	Pos	PosOpt	Neg	NegOpt	*BA3*
ACA	0	0	0	$\rightarrow 1$	0	0	1	1	1
ACC	1	1	0	0	0	$\rightarrow 1$	1	1	1
ACG	1	1	1	1	1	1	1	1	1
ACT	0	0	0	0	0	0	0	0	1
AGA	1	1	1	1	1	1	1	1	1
AGC	1	1	1	1	1	1	1	1	1
AGG	0	0	0	0	0	0	1	$\rightarrow 0$	0
AGT	0	0	0	0	0	0	1	1	1
ATA	0	0	0	0	0	0	0	0	0
ATC	1	1	1	1	1	1	1	1	1
ATG	0	0	1	1	1	1	1	1	1
ATT	1	1	0	0	0	0	0	0	0
CAA	0	$\rightarrow 1$	0	$\rightarrow 1$	0	$\rightarrow 1$	1	1	1
CAC	0	0	0	0	0	0	0	0	1
CAG	1	1	1	1	1	1	1	1	1
CAT	0	0	0	0	0	0	0	0	1
CCA	0	0	0	0	0	0	1	1	1
CCC	0	0	0	0	0	0	0	0	0
CCG	0	0	1	1	0	0	1	1	1
CCT	1	1	0	0	0	0	1	1	1
CGA	1	1	1	1	1	1	1	1	1
CGC	1	1	1	1	1	1	1	1	1
CGG	1	1	1	1	1	1	1	1	1
CGT	1	1	1	1	1	1	1	1	1
CTA	0	0	1	1	0	0	1	1	0
CTC	1	$\rightarrow 0$	0	0	0	0	1	1	1
CTG	1	1	1	1	0	0	1	1	1
CTT	0	0	0	0	0	0	0	0	1
GAA	1	1	1	1	1	1	1	1	1
GAC	1	1	1	1	1	1	1	1	1
GAG	1	1	1	$\rightarrow 0$	1	1	1	1	1
GAT	1	1	1	1	1	1	1	1	1
GCA	0	0	0	$\rightarrow 1$	0	$\rightarrow 1$	1	1	1
GCC	1	1	1	1	0	0	1	$\rightarrow 0$	1
GCG	1	1	1	1	1	1	1	1	1
GCT	1	1	1	1	1	1	1	1	1
GGA	1	1	1	1	1	1	1	1	1
GGC	1	1	1	1	1	$\rightarrow 0$	1	1	0
GGG	0	0	0	0	0	0	0	0	0
GGT	0	0	0	0	0	0	0	0	0
GTA	0	0	0	0	0	0	0	0	0

Continued on next page

Table 4.9: (continued)

Triplet	SP08	Opt08	SP09	Opt09	Pos	PosOpt	Neg	NegOpt	*BA3*
GTC	1	1	1	1	1	1	1	1	1
GTG	0	0	0	0	0	0	0	0	0
GTT	0	0	0	0	0	0	0	0	0
TAA	1	$\rightarrow 0$	0	0	0	0	0	0	0
TAC	0	0	0	0	0	0	0	$\rightarrow 1$	1
TAG	0	0	0	0	0	0	0	0	0
TAT	0	0	0	0	0	0	0	0	0
TCA	1	1	1	1	0	0	1	1	1
TCC	1	1	0	0	0	0	0	0	0
TCG	1	1	1	1	1	1	1	1	1
TCT	0	0	0	0	0	0	0	0	0
TGA	0	$\rightarrow 1$	1	1	1	1	1	1	1
TGC	0	0	0	0	0	0	1	1	1
TGG	1	1	1	1	1	1	1	1	1
TGT	0	0	0	0	0	0	0	0	0
TTA	0	0	0	0	0	0	0	0	0
TTC	1	1	1	1	1	1	1	1	1
TTG	0	0	0	0	0	0	0	0	1
TTT	0	0	0	0	0	0	0	0	0

Observing Table 4.9, we can get a feel for some (not all) triplets that may be important to the sequence discrimination process. First of all, notice that a change in a triplet conversion (due to the optimization process) to be in general agreement with the other models (BA3 best being the most important of all) should be a good clue that such a triplet conversion is significant to BA3 abstraction rules. For example, for SP '08, SP '09, and the Pos abstraction rule, the triplet conversion CAA $\rightarrow$ 0 is optimized to become CAA $\rightarrow$ 1 in agreement with the Neg, NegOpt, and BA3 best abstraction rules. Similarly, ACA, GCA, and ACC convert to be like the BA3 best conversions after various splicing potential maps were optimized. Given this information, might CpA or ApC be important signals to the classification process? Furthermore, TAA and TGA also convert to be like the BA3 best abstraction rule. These are both stop codons, but since we process the data in mixed frame their effect should not be strong. Not all optimizations produce an agreement though: the CTC triplet is optimized in SP '08 to be like the SP '09, SP'09 Optimized, Pos, and Pos Optimized maps; however, these do not match the Top 24 Negative Splicing Potential map (Neg), Neg Optimized, nor the BA3 best abstraction rule. All in all, only 3 of 13 triplets optimize to *disagree* with the BA3 best abstraction rule (CTC, GCC & GAG), suggesting that the BA3 best map is indeed a good classifier not only based on its relationship to splicing potential signals, but by its overall efficacy in sequence classification.

4.3.2 Abstraction rules for 1, 2, and 4-mers

One is not limited to the abstraction of DNA triplets for binary sequence generation, but can abstract single nucleotides, dinucleotides, tetranucleotides, and even longer DNA sequences. Any constraint on the choice of DNA abstraction level should depend both on the amount of available training data and the nature of the nucleotide signals being abstracted. We present the best as well as typical results for abstraction rules on the 1-mer, 2-mer, and 4-mer levels within this section. It is important to note that while the dataset being used is "mixed frame" (see Section 4.2.2), the notion of frame is irrelevant to abstraction rules outside of some multiple of three.

Table 4.10: The model accuracy of all 7 non-trivial BA1MM10 abstraction rules (abstracting single nucleotides, Markov chain order of 10).

Abstraction Rule	Exon Accuracy	Intron Accuracy	M-value
G-map	77%	79%	0.779
C-map	68	77	0.717
T-map	76	77	0.765
A-map	66	74	0.696
CG-map	70	73	0.717
GT-map	72	85	0.773
CT-map	82	73	0.771

The abstraction of single nucleotides into bits has already been shown in Figure 4-1 for the so-called "G-map" abstraction rule. A G-mapping abstraction rule means to convert all Gs to 1 and all As, Cs, and Ts to 0. Table 4.10 displays all abstraction rules for the single nucleotide binary abstraction level (BA1). The accuracy of each model for exon-intron classification is again given in terms of exon accuracy (percent correctly predicted exons) and intron accuracy (percent correctly predicted introns) from within the test set (see Table 4.1) as well as with our measure of overall model accuracy, the M-value which balances both intron and exon accuracy (see Section 4.2.3.1). The highest performing maps are the G-map and GT-map. This makes biological sense considering that G is the most common start of codons in the well-known RNY repetitive pattern of the genetic code [26]. Although our algorithm does not take into account the reading frame directly, the conversion of a nucleotide sequence to G versus nonG will still emphasize the 3 base-pair periodicity within coding sequences versus its lack within introns, which will easily be detected by the sliding window of the Markov chain algorithm. As for the strength of the GT-map—with its strong intron accuracy (85%, 6% better than any other single nucleotide mapping)—its efficacy is probably related to the GT-rich exonic splicing silencer signals that ought to be more prevalent within introns than exons due to their role as silencers [29].

One may object to only seven maps being referred to as the "complete" set of abstraction rules where one might expect 16 by intuition (2^4 possible nucleotide to bit conversions). One property of the abstraction process is that *inversions* are equivalent mappings within the abstraction rule space. Thus, a G-map is equivalent to an ACT-map and so-forth—the only difference being that each 1 and 0 exchanges places in the generated binary sequences. It is easy to see this will not change the classification of genomic sequences whatsoever. A second type of map removed was the removal of so-called "trivial abstraction rules" that convert all nucleotide patterns (of any length) to all 1s, or, alternatively, to all 0s. Such abstraction rules completely reduce the genomic information to zero—erasing all discriminative information in the primary sequences.

Table 4.11: The best abstraction rule for dinucleotides (BA2 best) with a comparison to the *A priori* 2 map (AP2). The frequencies (in percentage) of each dinucleotide is given for both CDS exons and introns. The percent difference between CDS and intron dinucleotide frequencies is also given.

BA2 best			*A priori* 2			
Dinucleotide		Abstraction	% in Exons	% in Introns	Diff.	AP2
AA	$\Rightarrow$	1	7.3%	8.7%	-1.4%	0
AC	$\Rightarrow$	1	5.5	4.7	$+0.9$	1
AG	$\Rightarrow$	1	8.1	7.0	$+1.1$	1
AT	$\Rightarrow$	1	5.3	7.5	-2.2	0
CA	$\Rightarrow$	1	8.0	6.8	$+1.2$	1
CC	$\Rightarrow$	1	7.6	5.2	$+2.4$	1
CG	$\Rightarrow$	1	3.2	1.1	$+2.1$	1
CT	$\Rightarrow$	1	7.0	7.2	-0.1	0
GA	$\Rightarrow$	1	7.6	5.8	$+1.8$	1
GC	$\Rightarrow$	1	6.9	4.4	$+2.5$	1
GG	$\Rightarrow$	0	7.0	5.4	$+1.6$	1
GT	$\Rightarrow$	0	4.5	5.5	-1.0	0
TA	$\Rightarrow$	0	3.2	6.5	-3.3	0
TC	$\Rightarrow$	0	5.7	5.9	-0.2	0
TG	$\Rightarrow$	1	7.9	7.7	$+0.2$	1
TT	$\Rightarrow$	0	5.1	10.7	-5.5	0

Moving on to abstraction rules for 2-mers or dinucleotides, we performed an exhaustive search of the abstraction rule space to find the best performing mapping using our validation set (see Section 4.2.3.3). For each 2-mer abstraction rule, the number of conversions taking place are $4^2 = 16$ abstractions. Thus, for the whole space one would need to test $2^{4^2} = 2^{16} = 65,536$ mappings. Discounting inversions and trivial mappings, we only need test $\frac{2^{16}}{2} - 2 = 32,766$ abstraction rules. The very best abstraction rule (BA2 best) is presented in Table 4.11. Dinucleotides starting with T (TpN) tend to be in the same conversion class (abstracting to 0), with the one exception of TpGs. GpGs also cluster with the TpN conversion class. Unfortunately, the two conversion classes represented by such an abstraction rule cannot reveal all that is going on biologically. The abstraction process happens on the short-range nucleotide level while the binary sequence itself will contain mid-range nucleotide patterns that the Markov model alone will be able to detect.

The problem becomes clearer when we compare the BA2 best abstraction rule to the *a priori* 2 (AP2) map as presented in Table 4.11. The AP2 map takes the percentage of every dinucleotide in both coding exons and introns and assigns 1 when the dinucleotide percentage is more abundant in exons, 0 if not. While the AP2 and BA2 best mappings are similar, they have some marked differences one would not predict based on the short-range dinucleotide abundances alone. For example, the ApT dinucleotide is 2.2% more abundant in introns (7.5%) than in exons (5.3%) and so is assigned to the 0 conversion class in AP2 but is in the 1 conversion class for the BA2 best mapping. The accuracies of the two models differ greatly too. The BA2 best map has an exon accuracy of 75%, an intron accuracy of 88%, and an M-value of 0.806 while the AP2 map has 74% accuracy for exons, 71% for introns, and an M-value of 0.726—quite a reduction for intron accuracy at 17% difference. As such it is reasonable to assume that nucleotide patterns longer than dinucleotides, which can only be detected by the Markov model on the binary sequence, are responsible for such a difference in accuracy. Moreover, the large change in intron accuracy suggests that these patterns may pertain to nucleotide signals in introns. The full accuracy of the BA2 best map is further elucidated in Table 4.15 where it is compared to BA4 maps and varied by Markov model order.

In order to find the BA4 best mapping we performed three optimization trials utilizing a hill-climbing method on a supercomputer (see Section 4.2.3.4 for details). Each trial started with an initial "seed" or starting value—a random map for trial 1, an arbitrary half 0/half 1 mapping (relative to a particular sorting of dinucleotides) for trial 2, and an arbitrary "striped" mapping with 0 and 1 alternating for each tetranucleotide. For each round a brute-force search of maps similar to the previous round's best abstraction rule was employed. Eventually the three trial maps converged to their maximal M-values, as is displayed in Table 4.12 for trial 1. The round convergence for trials 2 and 3 are shown in the chapter appendix, Tables 4.25 and 4.26 respectively. Trial 1 converged to the strongest M-value at 0.8543 after 30 rounds while trial 2 converged to an M of 0.8495 after 44 rounds and trial 3 to 0.8519 after 38 rounds of optimization. As one can see, each optimization trial produced a similar M-value, although trial 1 achieved the highest actual value within the landscape of classifier accuracies.

Table 4.13 shows the Hamming distances (δ) of each optimization trial best versus its peers. The largest possible value for a Hamming distance will be 256 conversions different (4^4 possible tetranucleotides) while the smallest possible value will be 0 for the same mapping compared against itself (displayed as a dash). Since we consider an inversion to be same map, as previously discussed, we can compare only the closest version of each map between itself and its inversions. Thus, the truly most distant mapping should be a Hamming distance of about 128 when considering inversions. It is interesting to see from Table 4.13 that each optimization trial best map is within about 50 conversions of each other map, pairwise. We believe this indicates a large plateau in the landscape of classifier accuracies where there exist many possible permutations of certain tetranucleotides between the two conversion classes. Unsurprisingly, this indicates that certain tetranucleotide patterns may be more important than others.

In order to analyze the difference between important and unimportant tetranucleotides, we show the conversion rules for each optimization trial best and display their consensus or agreement in Table 4.27 in fractional form. For the three BA4 optimization trial abstraction rules there are only two possible choices: 2 out of 3 maps agree, or all agree. To further our understanding, we also take the left and right dinucleotides from each 4-mer and compare the BA2 best mapping value to the mappings of the three BA4 optimization trial abstraction rules for the whole tetranucleotide.

Table 4.12: Optimization for 4-mer abstraction rules using a BA4MM12-like algorithm on the validation set. The optimal value appears in bold.

Seed	Exon Acc.	Intron Acc.	M-value	M-delta	HD
Round 1	57.8%	68.8%	0.6284	0.0000	0
Round 2	62.8	75.1	0.6834	0.0550	3
Round 3	65.2	77.7	0.7074	0.0240	6
Round 4	67.0	81.1	0.7311	0.0237	9
Round 5	67.9	84.0	0.7464	0.0153	12
Round 6	70.1	84.0	0.7601	0.0137	15
Round 7	71.2	85.2	0.7713	0.0112	18
Round 8	72.1	86.3	0.7802	0.0089	21
Round 9	73.1	87.0	0.7886	0.0084	24
Round 10	74.1	87.0	0.7951	0.0065	27
Round 11	74.4	88.2	0.8007	0.0056	30
Round 12	75.0	89.0	0.8070	0.0063	33
Round 13	75.6	89.4	0.8120	0.0050	36
Round 14	76.2	89.6	0.8164	0.0044	39
Round 15	76.4	90.7	0.8204	0.0040	42
Round 16	77.0	91.1	0.8256	0.0052	45
Round 17	77.4	91.8	0.8297	0.0041	48
Round 18	78.2	91.9	0.8352	0.0055	51
Round 19	78.5	92.8	0.8399	0.0047	54
Round 20	79.0	93.0	0.8430	0.0031	57
Round 21	79.4	92.8	0.8459	0.0029	60
Round 22	79.6	92.8	0.8472	0.0013	63
Round 23	79.7	93.2	0.8484	0.0012	66
Round 24	79.8	93.8	0.8502	0.0018	69
Round 25	80.1	93.4	0.8516	0.0014	72
Round 26	80.1	93.6	0.8521	0.0005	75
Round 27	80.1	94.0	0.8530	0.0009	78
Round 28	80.3	93.6	0.8536	0.0006	81
Round 29	80.4	93.4	0.8539	0.0003	84
Round 30	80.5	93.2	0.8540	0.0001	85
Round 31	**80.5**	**93.4**	**0.8543**	**0.0003**	**86**
Round 32	80.5	93.4	0.8540	-0.0003	89
Round 33	80.5	93.4	0.8543	0.0003	86

The agreement or consensus of all 5 nucleotide pattern conversions is also displayed in fractional form. For example, if AACC converts to 1 for the best abstraction from trial 1, converts to 0 for trial 2, and converts to 1 for trial 3, then the consensus would be a 2 out of 3 conversion agreement. Adding in the left and right dinucleotides of the aforementioned 4-mer (AA & CC respectively) the BA2 best map will convert AA to 1 while also converting CC to 1. The total consensus would be displayed as a 4 out of 5 conversion agreement. The minimum consensus will be either 3 out of 5, or, 2 out of 4. The case where there are only 4 total conversions occurs when the left and right dinucleotides are repeated in the tetranucleotide; such as with AGAG where the BA2 best map converts both of the AGs to 1. Clearly, this should be counted as only 1 conversion.

Table 4.13: The Hamming distance (δ) matrix for the three best BA4 optimization trials. The range of δ is $0 \leq \delta \leq 256$, since we are considering rules that convert all possible tetranucleotides.

$\delta(M_i, M_j)$	T1	T2	T3
BA4 Trial 1	-	51	44
BA4 Trial 2	51	-	49
BA4 Trial 3	44	49	-

We learn from Table 4.27 that extending the best abstraction rule from 2-mers to 4-mers is not straightforward. For example, AAAA converts unanimously to 0 for all three BA4 trials while BA2 converts AA to 1. The same is true for AAAT, AGCC, AGGC, ATAA, ATAT, CACA, CCCA, *et cetera.* Other DNA pattern conversions have perfect agreement between both BA4 and BA2 mappings: such as AAAG, AATG, ACCG, ATCA, and many others, suggesting the importance of these tetranucleotides to their conversion classes. Some tetranucleotides appear quite variable, like CCTA, where 2 out of 3 BA4 trial maps agree and 3 out of 5 maps agree after adding in the BA2 best conversions for CC and TA—suggesting the flexibility of assigning these tetranucleotides to either conversion class.

For each tetranucleotide in Table 4.27, we define L as the left dinucleotide of any fixed 4-mer and R as the right dinucleotide of the 4-mer. We use the expression "$L = R$" to indicate the event where the two dinucleotides of the given 4-mer have the same conversion (they agree) under the BA2 best abstraction rule. Let "*All*" indicate the event where complete (4 out of 4 or 5 out of 5) agreement between the BA4 trials

Table 4.14: The occurrences & probabilities for Table 4.27. *All* corresponds to a total consensus ($^5/_5$ or $^4/_4$), *Trials* corresponds to complete consensus for the BA4 optimization trials ($^3/_3$), and $L = R$ corresponds to agreement between the left and right dinucleotides of the BA4 tetranucleotide under the BA2 best abstraction rule.

Event	Occurrences	Probability
$L = R$	146	0.570
$L \neq R$	110	0.430
$Trials$	184	0.719
$\neg Trials$	72	0.281
All	86	0.336
$\neg$*All*	170	0.664
$All \cap L = R$	86	0.336
$\neg All \cup L \neq R$	170	0.664
$\neg Trials \cap L \neq R$	31	0.121
$Trials \cup L = R$	225	0.879
$L = R \cap Trials$	105	0.410
$L \neq R \cup \neg Trials$	151	0.590

and the BA2 dinucleotide conversions was reached, conversely, let $\neg$*All* mean that total agreement was not reached (2 out of 4, 3 out of 4, 3 out of 5, or 4 out of 5 conversions). We also use the event called "$Trials$" to indicate that only consensus within the three BA4 optmization trials was reached for a particular tetranucleotide (or not for $\neg Trials$). For example, for AAAG the $Trials$ event occurs since all three BA4 optimization trial abstraction rules convert AAAG to 1; second, $L = R$ occurs since both AA and AG are converted to 1 under the BA2 best abstraction rule; and, finally, the *All* event also occurs since we have a 5 out of 5 agreement among the various conversions for that tetranucleotide.

Table 4.14 shows both the occurrences and probabilities of these events, their negations, and the co-occurrences of certain events. We first ask the question of whether agreement in the dinucleotides L and R for each 4-mer within Table 4.27 indicates that we will have complete consensus (*All*) for the BA4 tetranucleotide mapping or not. In other words, does agreement in BA2 best map conversions indicate the same agreement under the three BA4 optimization trial map conversions for the tetranucleotide? We can answer this question with the conditional probability given Equation 4.7. Using Table 4.14 it is easy to compute:

$$\begin{aligned} Prob(All|L = R) &= \tfrac{Prob(All \,\cap\, L \,=\, R)}{Prob(L \,=\, R)} && (4.7) \\ &= \tfrac{.336}{.570} = 58.9\%, && (4.8) \end{aligned}$$

where $Prob(\cdot)$ is the probability of the given event. At a conditional probability of 58.9%, Equation 4.7 demonstrates that *a priori* knowledge of a BA2 mapping cannot tell us with great confidence how to map an abstraction rule on the 4-mer level. Next, we can ask the question of whether a lack of BA4 trial map consensus pertains strongly to a disagreement in the conversions of the dinucleotides under the BA2 best mapping or not. The case where the dinucleotides disagree will be denoted as $L \neq R$ and where the BA4 trials disagree will be denoted as $\neg Trials$. The probability of disagreement in the BA4 trials given disagreement in the BA2 dinucleotides is:

$$\begin{aligned} Prob(\neg Trials|L \neq R) &= \tfrac{Prob(\neg Trials \cap\, L \,\neq\, R)}{Prob(L \,\neq\, R)} && (4.9) \\ &= \tfrac{0.121}{0.430} = 28.1\%. && (4.10) \end{aligned}$$

We see from Equation 4.9 that *a priori* information of dinucleotide disagreement predicts that the trial maps will disagree for the tetranucleotide at a probability of only 28.1%. Thus, a lack of consensus (variableness in the conversions) of the BA4 trial maps for each tetranucleotide is not clearly related to a lack of agreement in the conversions (BA2 best) of dinucleotides that compose it. These two conditional probabilities taken together imply that finding strong tetranucleotide mappings (where there is complete consensus) versus weak ones (where the bit/conversion may be changed with little consequence) cannot be directly inferred from dinucleotide mapping but must be solved on the level of 4-mers; *i.e.*, through a search methodology. One may wonder about the conditional probability of $P(Trials \mid L = R)$. Given dinucleotide agreement, the trial consensus will indeed occur at a higher probability of 71.9% (still less than 90%); however, such a predicted trial consensus may be in consensus *against* the dinucleotide consensus of $L = R$ (e.g., where the two dinucleotides of BA2 are converted to 0 but the trials of BA4 all convert the tetranucleotide to 1). Hence, Equation 4.7 is a much better way of asking the question: does agreement in the dinucleotide mappings predict agreement (of the same kind) in the agreement of the tetranucleotide one?

Table 4.15 shows the exon accuracy, intron accuracy, M-value, and number of empty probabilities ($P_{\emptyset}$) for the three BA4 optimization trial best mappings as well as for the BA2 best abstraction rule. The Markov model order is varied from orders 1 to 15. Markov model order 1 tests (4 bp of effective nucleotide coverage for BA2MM1 and 8 bp effective nucleotide coverage for BA4MM1) already show high intron accuracy (91%, 93%, and 90% for trials 1 thru 3 respectively) for BA4 while the BA2 best mapping has a good but lower intron accuracy at 85%. Similarly, exon accuracy for the three BA4 trials are fair at 79%, 78% and 79% while the BA2 best map has a lower exon accuracy at 74% for Markov models of order 1. One may interpret the efficacy of these order 1 Markov models to mean that the binary-abstraction process itself is indeed extremely important to BAMM sequence classification accuracy.

The BA4 trials increase in M-value from an average of 0.838 at order 1 to peaking at Markov model order 9 for an average M of 0.848 to falling to an average of 0.813. Similarly, the BA2 best map goes from an M-value of 0.787 at MM1 to peaking at 0.808 at MM12 and falling to 0.749 at MM15. The reason for these trends is clear. Even on the short-range of order 1 Markov models the effective nucleotide coverage of BA4 at 8 bp and BA2 at 4 bp is sufficient for sequence discrimination, although

BA4 is better since it covers more nucleotide information. As the order increases so does the effective nucleotide coverage of the models, peaking at 40 bp for the BA4 trial maps (order 9) and at 26 bp for the BA2 best map (order 12). This implies that middle-range patterns (on the nucleotide level) may subtly improve the accuracy of the models just as increasing the order of a traditional homogeneous Markov model generally increases its accuracy as well. Unfortunately, the same limits of a traditional Markov model will surface for binary-abstracted Markov models: limited training data means that sequence patterns will have limited support and may not even occur at all within the training set. We call the occurrence of a sequence pattern that exists in the test set but not the training set an "empty probabilities" because that variable (a probability) is empty and has not been trained. In our current algorithm we ignore the empty probabilities in the Markov chain, but at what cost? As empty probabilities accumulate (over 49 thousand for BA2MM15) with respect to increasing the Markov model order, accuracy will inevitably suffer as more error is introduced. The only recourse is either to train with more data (sometimes infeasible) or lower the Markov model order until the empty probabilities disappear. For a more complete discussion of empty probabilities, see Section 4.2.5.1.

Table 4.15: The accuracy of BA2 and BA4 abstraction rules (the best for each optimization trials), varied by MM order. The data was evaluated on the test set. The number of occurrences of untrained observation in the test set (number of empty probabilities = $P_\emptyset$) is given in column 6.

MM Order	Model	Exon Acc.	Intron Acc.	M-value	$P_\emptyset$
Order 1	BA4 Trial 1	79%	91%	0.841	-
	BA4 Trial 2	78	93	0.837	
	BA4 Trial 3	79	90	0.837	
	BA2 Best	74	85	0.787	

Continued on next page

Table 4.15: (continued)

MM Order	Model	Exon Acc.	Intron Acc.	M-value	$P_\emptyset$
Order 2	BA4 Trial 1	80	91	0.844	-
	BA4 Trial 2	78	93	0.838	
	BA4 Trial 3	80	90	0.839	
	BA2 Best	74	85	0.791	
Order 3	BA4 Trial 1	80	91	0.845	-
	BA4 Trial 2	79	93	0.840	
	BA4 Trial 3	80	90	0.841	
	BA2 Best	75	85	0.793	
Order 4	BA4 Trial 1	80	91	0.845	-
	BA4 Trial 2	79	92	0.840	
	BA4 Trial 3	80	90	0.843	
	BA2 Best	75	86	0.795	
Order 5	BA4 Trial 1	80	91	0.847	-
	BA4 Trial 2	79	92	0.842	
	BA4 Trial 3	80	90	0.843	
	BA2 Best	75	86	0.796	
Order 6	BA4 Trial 1	80	91	0.846	-
	BA4 Trial 2	79	93	0.843	
	BA4 Trial 3	80	90	0.845	
	BA2 Best	75	87	0.800	
Order 7	BA4 Trial 1	80	91	0.847	-
	BA4 Trial 2	79	93	0.844	
	BA4 Trial 3	80	90	0.843	
	BA2 Best	75	87	0.800	
Order 8	BA4 Trial 1	80	91	0.847	-
	BA4 Trial 2	79	93	0.845	
	BA4 Trial 3	80	91	0.846	
	BA2 Best	75	87	0.802	
Order 9	BA4 Trial 1	80	91	0.849	-
	BA4 Trial 2	79	93	0.845	
	BA4 Trial 3	80	92	0.849	
	BA2 Best	75	88	0.805	
Order 10	BA4 Trial 1	80	92	0.849	-
	BA4 Trial 2	79	92	0.844	
	BA4 Trial 3	80	91	0.848	
	BA2 Best	75	88	0.806	
Order 11	BA4 Trial 1	80	92	0.849	0
	BA4 Trial 2	79	92	0.843	0
	BA4 Trial 3	80	90	0.843	0
	BA2 Best	76	88	0.807	1438

Continued on next page

Table 4.15: (continued)

MM Order	Model	Exon Acc.	Intron Acc.	M-value	$P_\emptyset$
Order 12	BA4 Trial 1	80	92	0.846	0
	BA4 Trial 2	79	92	0.842	0
	BA4 Trial 3	80	91	0.844	0
	BA2 Best	76	87	0.808	2469
Order 13	BA4 Trial 1	80	93	0.847	916
	BA4 Trial 2	79	92	0.840	0
	BA4 Trial 3	80	91	0.843	0
	BA2 Best	76	87	0.804	6480
Order 14	BA4 Trial 1	79	91	0.838	9529
	BA4 Trial 2	77	93	0.833	0
	BA4 Trial 3	79	92	0.839	729
	BA2 Best	76	86	0.806	19756
Order 15	BA4 Trial 1	78	82	0.799	42762
	BA4 Trial 2	75	93	0.817	0
	BA4 Trial 3	77	91	0.824	7882
	BA2 Best	77	73	0.749	49431

Binary-abstracted Markov models appear to be robust for a variety of binary-abstractions including on the 1, 2, and 4 nucleotide level. As the abstraction level increases so does the complexity of choosing the abstraction rule; however, we observe that increasing the abstraction level tends to yield better results. For BA1MM10, the G-map (a.k.a., the BA1 best) tested for an M-value of 0.779, BA2MM10 (BA2 best mapping) produced an M-value of 0.806, and BA4MM10 (BA4 trial 1, aka, BA4 best) yielded a still higher M-value of 0.849. Extending a BA2 best to BA4 best abstraction rule is not a straightforward process and we believe that finding the best mapping at each abstraction level requires an optimization strategy, even as we have undertaken (see Section 4.2.3.4). Now let us consider abstraction rules that do not requite a search strategy to obtain; rather, that rely on the nucleotide *context* to perform the abstraction.

4.3.3 Context-dependent Abstraction Methods

Insertions and deletions are common genomic events. However, the likelihood of an insertion or deletion (indel) being allowed to become fixed within a coding sequence is reduced because of the deleterious nature of frameshift mutations (as they may alter all downstream amino acid translations). In our large-scale investigation of indels in primates [9] we demonstrated that a large majority ($\sim 90\%$) of both short insertions and deletions (< 4bp) are associated with short homonucleotides runs (e.g., ggg, aaaa, or ccc). As we have said, coding sequences put strict restrictions on indels allowing only in-frame changes. Therefore, one could expect significant differences in the distribution and length of homonucleotide runs in exons versus introns.

The consequence of these biological patterns is that introns and exons should differ in terms of their repetitive structures. Although previous abstraction rules should be able to pick up these patterns, it makes sense to devise an abstraction rule that explicitly emphasizes repetitive signals. To do so, we have created context-dependent versions of our binary-abstracted Markov model. Two major classes of context-dependent binary-abstracted Markov models (CDBAMM) were created: one for emphasizing repetitive nucleotide patterns (duplication model) and one for emphasizing repetitive purine-pyrimidine signals. Figure 4-2 shows the nucleotide abstraction process for both of these models. Section 4.2.5 discusses these methods in detail.

In Table 4.16 we give the exon accuracy, intron accuracy, and M-value for various choices of window size and jump step for the duplication and purine-pyrimidine models. The duplication model always employs a Markov model of 4 while the purine-pyrimidine model uses MM6. The reason for this is to try to keep the Markov model order constant while maintaining good accuracy and avoiding a catastrophic number of empty probabilities across different window size and jump step parameters. Nevertheless, a number of parameter choices show higher than expected numbers of empty probabilities (such as window 2, step 1). A discussion of the empty probabilities problem is given in Section 4.2.5.1.

Table 4.16: The accuracy of context-dependent abstraction rules. BAMM4 is used for the Duplication rule; BAMM6 is used for Purine-Pyrimidine rule.

Abstraction Rule / Model		Exon Acc.	Intron Acc.	M-value
Window 1 Step 1	Duplication	78%	87%	0.818
	Purine-Pyrimidine	69	82	0.745
Window 2 Step 1*	Duplication	76	88	0.809
	Purine-Pyrimidine	67	78	0.717
Window 2 Step 2	Duplication	75	77	0.761
	Purine-Pyrimidine	69	68	0.685
Window 3 Step 1*	Duplication	74	80	0.772
	Purine-Pyrimidine	69	73	0.710
Window 3 Step 2	Duplication	78	73	0.756
	Purine-Pyrimidine	75	69	0.718
Window 3 Step 3	Duplication	51	71	0.597
	Purine-Pyrimidine	54	66	0.595
Window 4 Step 1*	Duplication	76	76	0.761
	Purine-Pyrimidine	71	65	0.678
Window 4 Step 2*	Duplication	81	69	0.743
	Purine-Pyrimidine	80	65	0.717
Window 4 Step 3	Duplication	31	67	0.462
	Purine-Pyrimidine	51	61	0.559
Window 4 Step 4	Duplication	87	64	0.729
	Purine-Pyrimidine	79	68	0.727

* These parameters yield a high number of empty probabilities in the test set ($P_\emptyset$).

The first observation one may make from Table 4.16 is that the duplication model (DUP) performs better, in general, than the purine-pyrimidine (YR) one. Second, the choice of window size 1, step 1 appears to be best performing duplication model (M-value of 0.818) as well as the best purine-pyrimidine result (M of 0.745). Intron accuracy (87% for DUP;82% for YR) is favored for this choice of parameters, similar to traditional BAMM models. Interestingly, for a parameter choice of window 4 step 4 sensitivity to exons (87% for DUP; 79% for YR) is greatly increased over intron identification. Thus redundant patterns must be discernible in the genetic code for tetranucleotides. A window of 3, step 3 produces a nearly random classification choice for exons ($\sim 50\%$) with mediocre intron accuracy; but, intriguingly, stepping by 1 or 2 instead of 3 (that is, by comparing some overlapping redundant patterns instead) produces both better exon and intron accuracy. In conclusion, while the efficacy of CDBAMM greatly depends on the choice of parameters and model, its use does not require searching a large abstraction rule space to generate good sequence classifiers. Moreover, CDBAMM emphasizes biologically relevant repetitive signals in the binary sequences that allows classification at a level exceeding the accuracy of the BA1 and BA2 best BAMM abstraction rules.

4.3.4 Combining Models using Machine-learning

Machine learning (ML) is a relatively new scientific field of study that strives to uncover the fundamental laws of learning by engineered systems and respond to the primary question of building computer systems that automatically improve their performance with experience [30]. Although it has close ties with both computer science and statistics, ML is distinct. In general, machine learning approaches become relevant for the solution of a given problem under the following circumstances: the problem is too complex for traditional programming based approaches; or, the domain is not understood well enough to be able to construct a closed-form mathematical model of the relationships among the variables of interest but there is substantial data to build relationships for those variables. The ML solution is expected to adapt to or customize for users or circumstances in the operational environment following deployment or fielding.

Machine learning has been applied to commercial or industrial systems that have been successfully fielded during the last two decades. A brief collection of application domains entail speech recognition, computer vision, recommendation systems, bio-surveillance, bioinformatics, robotics control, and drug discovery. Particularly in bioinformatics many difficult and challenging problems have been attempted through a variety of machine learning algorithms and techniques [31]. Problems from the bioinformatics domain with ML applications include phylogenetic tree construction in evolution, gene finding and motif identification in genomics, function prediction and structure prediction in proteomics and genomics, protein annotation in text mining and proteomics, gene annotation in genomics and proteomics, genetic networks in microarray and system biology, and microarray data pre-processing and data analysis.

Using multiple sources of evidence can be a way to increase predictive power in gene prediction. The so-called "Statistical Combiner" program uses the predictions of multiple gene-finders to do gene prediction [13]. This is accomplished with a ML algorithm called a decision tree and is similar in philosophy to what we will later show for our model combination process. Another technology used to combine multiple sources of evidence for gene prediction is the support vector machine (SVM). Accomplished gene finding programs such as mGene [12] (used on Nematodes) combine scores of different biological signal "detectors" to come up with plausible gene structures. In their case, such signals to be scored include donor and accept splice sites, translation initiation sites, transcription start sites, sequence length, *et cetera*. For our work we do not produce a sequence parse (the whole gene structure) and merely classify sequences according to their sequence composition, although in a full-fledged gene finding program we would want to explore how to exploit these biological signals as well in order to create a sequence parse.

Using a support vector machine we optimized our BAMM classifiers by redrawing the boundary line for positive and negative predictions. After all, zero may not be the optimal splitting point between the two prediction classes. Up until now, sequences predicted to be an exon will have a positive log-likelihood (computed using the Markov chain) while introns will have a negative score (see Section 4.2.1.3 for more details). An SVM can redraw the boundary line not only linearly, but with respect to higher dimensional spaces. The solution will be robust in that it maximizes the distance between the two prediction classes [32]. In order to optimize the classifier, the SVM is trained using the output prediction scores for each tested sequence rather than the actual binary sequence data. Any refinement in the classification process will therefore be based on the preliminary predictions of the model. The SVM then tries to come up with a boundary that splits the two classes optimally while avoiding overtraining. Full details of how we train and apply the SVM technology is described in Section 4.2.6.

SVM optimization for a diverse selection of abstraction rules are shown side by side with their original model predictions in Table 4.17. For all BAMM models, Markov model 10 was used as the standard for the binary sequences. The SVM itself used a polynomial kernel of degree 3 to transform the data into a higher dimensional space. The non-homogeneous kernel was also used with data normalization turned on for better accuracy and quicker solution convergence. For each and every map some increase in M-value was reached due to the optimization process: a minimum of +0.001 for the homogeneous nucleotide Markov model of order 6 to a maximum of +0.091 for the *a priori* 3 (AP3) BAMM abstraction rule. This demonstrates the ability of the SVM to improve the accuracy of single classifiers.

Another observation is that the optimization technique typically comes with a trade-off (with the exception of AP3 and the purine-pyrimidine model). For the optimization to work, a few points of intron accuracy are “spent” in order to “earn” a few extra points of exon accuracy. This trade-off between sensitivity and specificity is a common phenomena when trying to optimize a classifier without adding new information. For example, for the GT-rich BA3MM10 model, 13 percentage points of intron accuracy are lost in order to gain 29 points in exon accuracy. Clearly this trade-off is desirable. In the case of AP3 and the YR model, both intron and exon accuracy increase but exon accuracy increases more. For AP3 17 points of exon accuracy are gained while only 6 points of intron accuracy are gained, similarly, for

Table 4.17: A diverse selection of abstraction rules are shown with their original accuracies versus their SVM optimization. All models except the traditional nucleotide Markov model 6 use Markov model order 10 as the standard value. The SVM used a non-homogeneous polynomial kernel of degree 3 with normalization.

	Original Values			SVM-optimized		
Abstraction Rule	% Exons	% Introns	M-val	%Exons	% Intron	M-val
Markov Model 6	89%	83%	0.854	94%	80%	0.855
G-map (BA1)	77	79	0.779	94	72	0.801
BA2 Best	75	88	0.806	94	81	0.860
BA3 Best	77	93	0.831	94	86	0.893
BA4 Best	80	92	0.849	95	84	0.883
A priori 3	76	69	0.726	93	75	0.817
SP Top 24 Pos	73	86	0.782	94	76	0.822
GT-rich	65	83	0.725	94	70	0.781
Duplication	77	86	0.807	95	76	0.829
Purine-pyrimidine	79	65	0.707	93	69	0.777

the YR model, 14 percentage points of exon accuracy are gained while only 4 points of intron accuracy are gained.

One may ask why exon is accuracy preferred. We believe that because of the diversity (both by size and composition) of introns versus exons (which are greatly constrained by the genetic code) that the log probability ratios or score values which the SVM uses for training and testing have a much larger spread for introns than exons and therefore introns are harder to optimize. Here are the facts: the average size of exons for our CDS dataset is 159.7 bp while the average size of our intron dataset is 5,423 bp (see Table 4.1). For each of the 10 models listed in Table 4.17 we took the score values of the introns and exons used to train the SVM (2,484 elements in each group) and calculated the standard deviations within both the exon and intron groups. Within introns, the largest standard deviation of the score values (predictions) was 991.2 for Markov model 6 while the lowest was 38.1 for the GT-rich BA3MM10. The average standard deviation across all 10 classifiers was 235.54 for introns. By contrast, the highest standard deviation for exon score values was 20.5 (MM6) while lowest was 1.4 (DUP) with a mean standard deviation of 4.45 across

Table 4.18: Using a support vector machine (non-homogeneous polynomial kernel degree 3 with data normalization) we combine K out of the $N = 10$ classifiers as listed in Table 4.17 and select the "best" classifier combination for each fixed $K = 1, 2, \ldots, 10$. The accuracy using the validation set is given versus the accuracy under the test set. The "best" model combination for each group must have the largest M-value in the test set and will differ from the M-value of the validation set by no more than 0.003.

		Validation Set			Test Set		
$\binom{n}{k}$	Best Combo	%ex	%in	M-val	%ex	%in	M-val
$K = 1$	BA3 Best (BA3)	93.9	86.6	0.896	93.9	86.2	0.893
$K = 2$	BA3 + BA4	94.7	89.2	0.915	94.5	89.8	0.918
$K = 3$	BA1 + BA3 + AP3	94.8	91.9	0.932	94.6	92.2	0.933
$K = 4$	MM6 + BA3 + BA4 + AP3	97.1	92.7	0.944	96.6	93.3	0.947
$K = 5$	MM6 + BA3 + BA4 + AP3 + DUP	96.8	93.6	0.950	96.5	94.0	0.951
$K = 6$	MM6 + BA2 + BA4 + AP3 + POS + DUP	96.6	93.8	0.950	96.3	94.4	0.952
$K = 7$	MM6 + BA2 + BA3 + BA4 + AP3 + POS + DUP	96.9	94.3	0.954	96.7	94.7	0.956
$K = 8$	MM6 + BA2 + BA3 + BA4 + AP3 + POS + GT + DUP	96.7	94.7	0.956	96.6	95.2	0.958
$K = 9$	MM6 + BA2 + BA3 + BA4 + AP3 + POS + GT + DUP + YR	96.6	94.8	0.956	96.3	95.1	0.956
$K = 10$	All Models	96.6	94.6	0.954	96.4	94.5	0.954

all classifiers. Thus, because the spread of scores was much smaller for exons, the SVM will be biased toward exon accuracy over intron accuracy. We chose to use the standard deviation for this analysis rather than the coefficient of variation since SVM training uses a geometric interpretation of the data that will be dependent on the scales used.

Notice also from Table 4.17 that the unoptimized homogeneous nucleotide Markov model of order 6 is more exon accurate (89% for exons versus 83% for introns) while the unoptimized BA3 best map is more intron accurate (93% accuracy for introns versus 77% for exons). This complementarity in accuracy makes one hopeful for combining different classifiers *together* to produce even better results (as well as to emphasize BAMM's efficacy in intron identification). Table 4.18 shows the best combinations under the polynomial kernel SVM for some choice of K classifiers/models out of $N = 10$ models. For N, Table 4.17 lists the choice of 10 models along with their SVM optimized versus non-optimized accuracies. Clearly, for each fixed $K = 1, 2, 3, \ldots, 10$ there will be a different number of combinations being explored. For $\binom{10}{10}$ only 1 combination is possible while for $\binom{10}{5}$, 252 combinations must be analyzed. The validation set (see Table 4.2) is used to optimize individual models, such as in the search for the BA3 best map. The test set is a disjoint dataset used for the final say on accuracy, although one can still employ the validation set as a second standard of accuracy for comparison. If the difference in goodness or M-value is small between the two datasets we can assume some robustness in the particular choice of model(s). Moreover, a small difference between the two datasets implies that there is not much over-fitting due to our abstraction rule optimization process. Thus, we limit any variation in M-value to no more than a 0.003 difference between the test and validation sets. The model combination for each fixed K with the largest M-value, also satisfying our condition for robustness, is considered the "Best Combo" for that group.

From Table 4.18 we notice that by combining models using the SVM accuracy does tend to increase. For $K = 1$ we start at our best individually optimized classifier with an M equal to 0.893. The M-value shoots up quickly from $K = 1$ to $K = 4$ (0.893 to 0.947) and then slows down to the peak value at $K = 8$ (0.958). After that adding additional classifiers only subtly decreases the M-value. To balance the number of models being combined (complexity) with accuracy concerns, we recommend $K = 5$ as a practical number of classifiers with very good accuracy (0.951) as well as good robustness (0.001 difference).

Throughout the tests, MM6, BA3, BA4, AP3, and DUP appear more than the other classifiers (at least 6 out of 10 times) within each best combination grouping. This indicates that the maps chosen in the $K = 5$ grouping are consistently complementary when in combination with each other. Moreover, notice that the top 7 optimized individual mappings from Table 4.17 include our set from $K = 5$. Also in that top 7 is the BA2 best map. Although the BA2 and BA4 best maps do not abstract the *same* information, as we have already discussed (Section 4.3.2), it may be close enough to pick BA4 over BA2 in this particular case since BA4 has the larger M-value. Why the slightly better splicing potential map is not picked for the best combination versus the *a priori* 3 map is unknown except that we hypothesize that splicing signals may also be emphasize in the BA3 and BA4 mappings. Whatever the case, it is clear that an SVM (along with other machine-learning methods [data not shown]) can powerfully combine multiple classifiers to produce even better results in sequence discrimination.

4.3.5 Using frame information.

Up until now we have only explored our abstraction rules using mixed frame datasets. We now present data for maps that depend on the reading frame. The same BAMM algorithm is applied as before only that we ensure uniformity in the reading frame for both the training and testing phase of the method. Note that we have not constructed an inhomogeneous Markov model, but rather, have merely restricted the training and test data (in effect dividing the non-homogeneous method into three homogeneous instances).

We obtained the codon usage patterns for the human genome from the Codon

Usage database [33]. From this information we formulated four different abstraction rules based on codon usage and codon bias patterns. Codon bias is relative to the amino acid and the first two codon positions while codon usage is relative to the codon frequency over the whole table of triplets. Table 4.19 shows the maps for abstraction rules emphasizing high (↑) and low (↓) codon bias as well as high and low codon usage (top 23 most frequent and top 23 least frequent codons). The biological feature that has emphasis will be a part of the 1-conversion class (mapped to 1).

Table 4.19: Abstraction rules based on codon bias and usage. The first column shows the amino acid while the second shows the codon usage frequency value. The last four columns give the frame-dependent maps that emphaize high and low codon bias followed by high and low usage respectively. Codon bias frequency is determined relative to the individual amino acids/codon positions 1 & 2 while codon usage frequency is relative to the whole table of codons.

AA	Codon	Usage Freq.	↑ c. bias	↓ c. bias	↑ c. usage	↓ c. usage
Ala	GCG	0.007	0	1	0	1
	GCA	0.016	0	0	0	0
	GCT	0.018	0	0	1	0
	GCC	0.028	1	0	1	0
Arg	CGT	0.005	0	1	0	1
	CGA	0.006	0	0	0	1
	CGC	0.010	1	0	0	1
	CGG	0.011	0	0	0	1
	AGG	0.012	1	0	0	1
	AGA	0.012	0	0	0	1
Asn	AAT	0.017	0	1	0	0
	AAC	0.019	1	0	1	0
Asp	GAT	0.022	0	1	1	0
	GAC	0.025	1	0	1	0
Cys	TGT	0.011	0	1	0	1
	TGC	0.013	1	0	0	0

Continued on next page

Table 4.19: (continued)

AA	Codon	Usage Freq.	↑ c. bias	↓ c. bias	↑ c. usage	↓ c. usage
Gln	CAA	0.012	0	1	0	0
	CAG	0.034	1	0	1	0
	GAA	0.029	0	1	1	0
	GAG	0.040	1	0	1	0
Gly	GGT	0.011	0	1	0	1
	GGG	0.016	0	0	0	0
	GGA	0.016	0	0	0	0
	GGC	0.022	1	0	1	0
His	CAT	0.011	0	1	0	1
	CAC	0.015	1	0	0	0
Ile	ATA	0.007	0	1	0	1
	ATT	0.016	0	0	0	0
	ATC	0.021	1	0	1	0
Leu	CTA	0.007	0	1	0	1
	TTA	0.008	0	1	0	1
	TTG	0.013	1	0	0	0
	CTT	0.013	0	0	0	0
	CTC	0.020	0	0	1	0
	CTG	0.040	1	0	1	0
Lys	AAA	0.024	0	1	1	0
	AAG	0.032	1	0	1	0
Met	ATG	0.022	1	0	1	0
Phe	TTT	0.018	0	1	1	0
	TTC	0.020	1	0	1	0
Pro	CCG	0.007	0	1	0	1
	CCA	0.017	0	0	0	0
	CCT	0.018	0	0	1	0
	CCC	0.020	1	0	1	0
Ser	TCG	0.004	0	1	0	1
	AGT	0.012	0	1	0	1
	TCA	0.012	0	0	0	0
	TCT	0.015	0	0	0	0
	TCC	0.018	1	0	1	0
	AGC	0.019	1	0	1	0
Thr	ACG	0.006	0	1	0	1
	ACT	0.013	0	0	0	0
	ACA	0.015	0	0	0	0
	ACC	0.019	1	0	1	0
Trp	TGG	0.013	1	0	0	0

Continued on next page

Table 4.19: (continued)

AA	Codon	Usage Freq.	↑ c. bias	↓ c. bias	↑ c. usage	↓ c. usage
Tyr	TAT	0.012	0	1	0	1
	TAC	0.015	1	0	0	0
Val	GTA	0.007	0	1	0	1
	GTT	0.011	0	0	0	1
	GTC	0.014	0	0	0	0
	GTG	0.028	1	0	1	0
Stop	TAG	0.001	0	1	0	1
	TAA	0.001	0	1	0	1
	TGA	0.002	0	1	0	1

The first and most important observation is that codon usage and codon bias are *similar*, not equivalent. For example, the same conversions are observed under glycine for high and low codon bias versus high and low codon usage maps respectively, but this trend is definitely not always the case. Sometimes the most significant codon of an amino acid is still rare, by and large, among the set of all possible codons (e.g., the TAC codon of tyrosine). The converse is also true, as is the case for the AAA codon of lysine (considered low under codon bias but definitely a highly used codon overall). Sometimes multiple codons are selected for the high and low codon bias maps, as is the case for leucine. The principal reason for this is because the amino acid has enough codons such that the first two codon positions may differ. The most frequent and last frequent of each subgroup (by the first two codon positions) is used for the high or low codon bias maps respectively. There are other exceptions. All of the stop codons are selected to be in the low codon bias mapping since they are expect to occur only once per coding sequence (in frame). However, this choice naturally balances emphasizing one-codon amino acids like methionine and tryptophan for the high codon bias map.

Table 4.20 presents the model accuracy high and low codon bias, high and low codon usage, and all possible combination of stop codons. Long open reading frames are an important property of coding exons sometimes exploited in gene finding algorithms in the early rounds of *ab initio* self-training [6]. By emphasizing some choice of stop codons [stop codon(s) converted to 0, else 1] we can use open reading frame-like patterns to do sequence classification. This is nothing new, although it does demonstrate the extremes of frame-dependent mappings under our method. Results are presented in frame as well as frame-shifted by 1 and 2 nucleotides; the results are also varied by Markov model order. The training and test dataset statistics are given Table 4.2.

Table 4.20: Model accuracy for frame-dependent abstraction rules. The accuracy for each abstraction rule is given in all three reading frames and varied by Markov model order. The ↑ refers to maps emphasizing something high while the ↓ are for maps emphasizing something low. Codon bias frequency is determined relative to the individual amino acids while codon usage frequency is relative to the whole table of codons.

Parameters		In frame			Frameshift 1			Frameshift 2		
MMk	Map	%ex	%in	M	%ex	%in	M	%ex	%in	M
	↑ cod. bias	59	79	0.673	73	73	0.728	61	60	0.607
	↓ cod. bias	66	67	0.667	74	70	0.716	63	52	0.574
	↑ cod. usage	72	89	0.789	52	55	0.530	60	70	0.645
	↓ cod. usage	77	83	0.802	60	56	0.579	55	65	0.599
	taa/tga	94	90	0.920	58	58	0.579	45	75	0.575
$k = 2$	tga	95	88	0.909	49	63	0.555	54	89	0.663
	taa	97	82	0.870	73	63	0.679	66	52	0.581
	tag/tga	94	92	0.929	67	43	0.531	46	74	0.574
	taa/tag/tga	94	92	0.927	59	62	0.603	22	72	0.416
	taa/tag	96	87	0.906	67	68	0.677	67	63	0.649
	tag	98	86	0.897	69	61	0.649	79	71	0.749

Continued on next page

Table 4.20: (continued)

Parameters		In frame			Frameshift 1			Frameshift 2		
MMk	Map	%ex	%in	M	%ex	%in	M	%ex	%in	M
$k = 4$	↑ cod. bias	57	79	0.665	74	73	0.732	59	62	0.607
	↓ cod. bias	65	67	0.661	74	71	0.722	68	53	0.595
	↑ cod. usage	72	88	0.787	51	59	0.546	60	70	0.644
	↓ cod. usage	77	83	0.801	61	56	0.581	57	65	0.607
	taa/tga	94	90	0.918	56	59	0.578	41	76	0.549
	tga	95	88	0.908	47	64	0.548	53	89	0.659
	taa	97	82	0.870	73	63	0.679	66	51	0.576
	tag/tga	94	92	0.931	51	55	0.525	41	77	0.551
	taa/tag/tga	94	92	0.928	58	63	0.604	47	67	0.562
	taa/tag	96	88	0.908	66	69	0.674	66	63	0.646
	tag	98	86	0.897	69	62	0.652	79	71	0.747
$k = 6$	↑ cod. bias	56	80	0.663	74	73	0.738	60	66	0.632
	↓ cod. bias	64	68	0.662	74	71	0.724	70	53	0.608
	↑ cod. usage	72	89	0.786	51	63	0.567	60	72	0.658
	↓ cod. usage	77	83	0.800	62	56	0.587	57	65	0.604
	taa/tga	94	90	0.917	57	59	0.584	39	78	0.541
	tga	95	88	0.909	48	62	0.547	53	89	0.659
	taa	97	82	0.870	73	64	0.681	65	51	0.576
	tag/tga	94	92	0.930	60	52	0.559	38	77	0.536
	taa/tag/tga	93	92	0.927	58	63	0.605	49	67	0.572
	taa/tag	96	88	0.907	66	68	0.672	66	63	0.641
	tag	98	86	0.897	69	62	0.653	79	72	0.752
$k = 8$	↑ cod. bias	56	83	0.663	75	74	0.745	62	70	0.655
	↓ cod. bias	64	69	0.666	75	71	0.727	70	57	0.629
	↑ cod. usage	72	89	0.784	52	65	0.578	60	72	0.654
	↓ cod. usage	77	84	0.799	61	58	0.593	56	66	0.606
	taa/tga	94	90	0.918	56	60	0.581	37	80	0.534
	tga	95	88	0.908	46	63	0.537	52	89	0.651
	taa	97	81	0.866	73	64	0.684	65	52	0.580
	tag/tga	94	92	0.929	57	54	0.551	35	80	0.517
	taa/tag/tga	94	93	0.931	57	65	0.605	51	68	0.586
	taa/tag	96	88	0.907	65	68	0.665	65	63	0.638
	tag	98	85	0.896	69	62	0.654	79	72	0.756

Continued on next page

Table 4.20: (continued)

Parameters		In frame			Frameshift 1			Frameshift 2		
MMk	Map	%ex	%in	M	%ex	%in	M	%ex	%in	M
	↑ cod. bias	55	83	0.664	75	75	0.749	64	68	0.658
	↓ cod. bias	64	70	0.669	75	71	0.727	69	60	0.642
	↑ cod. usage	72	89	0.786	54	65	0.593	62	72	0.666
	↓ cod. usage	76	84	0.798	62	59	0.606	54	69	0.606
	taa/tga	94	90	0.918	56	60	0.578	36	81	0.528
$k = 10$	tga	95	88	0.909	45	63	0.531	51	90	0.644
	taa	97	81	0.866	73	64	0.683	65	52	0.578
	tag/tga	94	92	0.931	57	53	0.546	33	83	0.513
	taa/tag/tga	94	93	0.933	56	64	0.595	50	67	0.579
	taa/tag	96	87	0.906	64	68	0.661	64	63	0.634
	tag	98	85	0.896	69	63	0.657	79	71	0.750
	↑ cod. bias	56	83	0.665	74	76	0.749	65	67	0.658
	↓ cod. bias	64	73	0.682	75	72	0.733	66	64	0.650
	↑ cod. usage	72	88	0.783	55	65	0.599	63	71	0.668
	↓ cod. usage	77	83	0.794	64	59	0.615	51	67	0.582
	taa/tga	94	90	0.916	55	60	0.575	35	82	0.524
$k = 12$	tga	95	88	0.909	43	65	0.526	49	90	0.633
	taa	97	81	0.863	73	64	0.682	64	52	0.577
	tag/tga	94	92	0.930	56	55	0.553	32	83	0.503
	taa/tag/tga	94	93	0.934	55	62	0.588	49	69	0.578
	taa/tag	96	87	0.905	64	68	0.659	63	62	0.628
	tag	98	86	0.897	69	64	0.662	79	71	0.749

It is important to remember that the results presented are both trained and tested in the same, uniform reading frame—rather than being trained in mixed frame and then tested in mixed reading frame. Since training and testing are done in the same frame, fluctuation in model accuracy by reading frame demonstrates the difference in genomic signals that the abstraction rule is able to emphasize within each frame. For the stop codon TGA-map using Markov model order 2, high sensitivity to exons and introns (95% and 88% respectively) are observed in frame since coding exons will contain very few TGA stop codons; however, abstracting coding exons in frames 1 and 2 incur a significant drop in accuracy (M of 0.555 and 0.663 respectively) since the binary sequences of coding exons will look very similar to those of introns under the stop codon abstraction rule.

Codon bias maps are most accurate within a frameshift of 1 while codon usage maps favor in frame data. In other words, the codon usage abstraction rule can emphasize the coding signals best within the open reading frame while the codon bias abstraction rule emphasizes signals best within data frame-shifted by 1. Increasing the Markov model order generally increases accuracy, but not so for all models in all frames. The best result from Table 4.20 is the taa/tag/tga-map at MM12 with an exon accuracy of 94% and and intron accuracy of 93% (M-value of 0.934) within frame—a gain from Markov model order 2 (M-value of 0.927)—but the accuracy actually decreases within frameshift 1 dataset (from M-value $= 0.603$ at MM2 to M-value $= 0.588$ at MM12) indicating the actual binary sequences are truly similar between coding exons and introns.

Table 4.21 provides a basis for comparison with Table 4.20 by selecting other mappings not generally believed to be frame-dependent for biological reasons. Interestingly, the degree to which each abstraction rule can emphasize biological signals does differ slightly from frame to frame and from rule to rule. The AG-rich map extracts the most difference between coding exons and introns under a frameshift of 1 while the GT-rich map is the most accurate within the frame. The BA3 best abstraction rule was optimized under the conditions of a mixed frame validation set (see Section 4.2.2) where the percentage of in frame data exceeded naturally occurring frameshift 1 data which exceeded naturally occurring frameshift 2 data. Accordingly, the BA3 best map is slightly more accurate within frame (M-value of 0.843) than in frameshift 1 ($M = 0.830$) and also more accurate under frameshift 1 than under a frameshift of 2 $M = 0.819$. However, since intron phases 1 and 2 are "swapped" for the coding exons of the mixed dataset we cannot conclude that the reading frame alone is responsible for the fluctuation in accuracy, but perhaps, some slight difference in the genomic signals of coding exons interrupted at each intron phase.

Table 4.21: Analysis of BA3MM10 abstraction rules in all three reading frames. Exon and intron accuracy is denoted as %ex or %in respectively. The M-value or measure of goodness is also given for each reading frame.

	In frame			Frameshift 1			Frameshift 2		
Abstraction Rule	%ex	%in	M	%ex	%in	M	%ex	%in	M
GC-richness	71	68	0.693	68	67	0.675	68	67	0.672
GT-richness	65	86	0.734	67	86	0.748	65	85	0.732
AG-richness	68	76	0.720	69	69	0.685	69	73	0.711
BA3 Best	79	94	0.843	79	89	0.830	76	90	0.819
Apriori 3	76	74	0.752	77	73	0.751	78	71	0.745
SP version 2008	72	88	0.787	66	71	0.683	65	73	0.685
SP '08 Optimized	75	88	0.805	70	74	0.721	74	81	0.772
SP version 2009	73	86	0.785	75	79	0.770	72	79	0.753
SP '09 Optimized	76	87	0.809	77	85	0.806	74	87	0.795
SP '09 Top 24 Pos.	75	86	0.801	72	78	0.747	72	81	0.761
SP '09 Positive Opt.	77	88	0.817	74	84	0.783	74	86	0.790
SP '09 Top 24 Neg.	77	77	0.771	77	78	0.779	78	76	0.768
SP '09 Negative Opt.	79	87	0.827	80	83	0.813	77	83	0.801

The *a priori* 3 map shows the most uniform accuracy across all reading frames; that is, the standard deviation of the M-values is 0.004. This makes sense since the triplet frequencies within coding exons and introns used to generate the abstraction rule were reckoned in a frameless manner. Importantly, the splicing potential maps tend to decrease in their M-value standard deviation across all three frames when they are optimized—with the exception of the Top 24 negative splicing potential map—suggesting that optimizing maps for model accuracy should not encourage any sort of abstraction rule frame-dependence within the context of a mixed frame validation dataset.

Whatever the case, the M-value standard deviation of the BA3MM10 best abstraction rule is only 0.012 while the very *smallest* M-value standard deviation for Table 4.20 (MM10) is 0.043 for the low codon bias abstraction rule. The average (per model) M-value standard deviation for Table 4.20 (MM10) is 0.142 with a maximum M-value standard deviation of 0.232 achieved by the stop codon map for TAG/TGA. All this indicates that while some models, like the BA3 best and AP3 maps operate in a nearly frame-independent manner, others models, such as those created by the stop codon maps, are extremely frame-dependent.

4.3.6 Applications to untranslated regions.

We now turn from discriminating introns and coding exons to exploring sequence classification versus untranslated exons instead. Untranslated exons are spliced, just like coding exons, at both the $5'$ and $3'$ ends, although $5'$ exons are more abundant in the currently annotated human genome. Untranslated regions (UTRs) are more difficult to find than coding sequences because they lack the structure and periodicity of the genetic code. However, splicing signals should still be retained for proper mRNA processing to occur (via exonic splicing enhancer sequences).

Unfortunately, fewer $5'$ and $3'$ UTR exons are available for training and testing versus coding exons. Table 4.1 demonstrates that there are over 3.5 times as many CDS exons available as either $5'$ or $3'$ UTR exons combined)—although the actual sequence size of $3'$ UTRs can be very great (an average of 829.4 bp). In order to deal with this dearth of data, we train our models using CDS exons and test them on both $5'$ and $3'$ UTR exons. Such a result will show the amount of similarity between the coding versus untranslated exons, although one should normalize by the difficulty of discriminating versus introns in the first place for that particular model. UTR introns are also employed in order to gain the most precise predictions ($5'$ or $3'$ UTR introns respectively). Our definitions for untranslated exons and introns are given in Section 4.2.2 and random dataset statistics are shown in Table 4.2.

In Table 4.22 we show accuracy of BA3MM10 abstraction rules that were trained on the normal dataset (CDS exons & all introns) and tested on all of the $5'$ UTR exons and introns (Table 4.1; in the second part of the table we show the effect of trying to train and test with $5'$ UTR random datasets—a much smaller dataset (Table 4.2).

Table 4.22: The accuracy of BA3MM10 maps on $5'$ UTR data. In the first part models are trained as usual on the normal training dataset of coding sequences and all introns but tested on the whole dataset of $5'$ UTRs (Table 4.1). In the second part random datasets are used for both training and testing (Table 4.2).

Trained on CDS exons & all introns, tested on $5'$ UTR exons & introns.			
Abstraction Rule / Model	Exon Accuracy	Intron Accuracy	M-value
GC-richness	82%	61%	0.700
GT-richness	52	88	0.650
AG-richness	59	73	0.655
BA3 Best	66	93	0.757
A priori 3	85	68	0.749
SP version 2008	76	74	0.748
SP '08 Optimized (HD4)	76	79	0.775
SP version 2009	78	76	0.770
SP '09 Optimized (HD4)	77	81	0.787
SP '09 Top 24 Positive	76	76	0.758
SP '09 Positive HD4	72	82	0.766
SP '09 Top 24 Negative	82	73	0.773
SP '09 Negative HD4	76	84	0.794
Trained and tested on $5'$ UTR data for introns & exons.			
Abstraction Rule / Model	Exon Accuracy	Intron Accuracy	M-value
GC-richness	68%	69%	0.689
GT-richness	55	66	0.604
AG-richness	60	58	0.592
BA3 Best	71	81	0.755
A priori 3	73	71	0.719
SP version 2008	72	73	0.721
SP '08 Optimized (HD4)	73	73	0.730
SP version 2009	70	82	0.752
SP '09 Optimized (HD4)	72	79	0.752
SP '09 Top 24 Positive	69	79	0.736
SP '09 Positive HD4	71	79	0.745
SP '09 Top 24 Negative	75	71	0.729
SP '09 Negative HD4	75	79	0.767

When we compare the first part of Table 4.22 to Tables 4.6 and 4.7 we observe that there is a loss in intron accuracy for unoptimized splicing potential maps at a slight gain of exon accuracy. A general loss of accuracy should be expected you train and test with mismatching groups of sequences. The slight improvement in exon accuracy suggests that splicing signals within CDS exons are able to be used to help detect 5′ UTR exons, since that is what these maps are emphasizing. As usual, the optimized versions of these maps improve intron accuracy, but instead of the slight increase in exon accuracy, as shown in Table 4.7, the exon accuracy may decrease a bit. Interestingly, the *a priori 3* and GT-rich gain significant exon accuracy percentage points (9% and 14% respectively) when compared to their CDS-trained/CDS-tested counterparts while intron accuracy remains stagnant or is reduced slightly (-1% and -4% respectively). This suggests that these maps must emphasize splicing signals too (exon splicing silencers are often GT-rich [29] and AP3 is built by looking at the triplets frequencies, not codon frequencies, in introns and CDS exons).

On the other hand, the BA3 best abstraction rule remains adept at identifying introns (at 93%) with no net loss in accuracy compared to CDS-trained/CDS-tested results. We conjecture that the BA3 best map may be optimized to detect other genomic signals than splicing ones in order to correctly identify introns. However, exon accuracy does decrease when used on 5′ UTR exons (-11%) indicating that the BA3 best abstraction rule may also be emphasizing the periodicity of the genetic code or some such related pattern unique to CDS exons. When we train and test using 5′ UTR data in the second part of Table 4.22, some of these interesting features are lost (such as increase in GT-rich exon accuracy). We believe that the small sizes for the random datasets are prohibitively small for a fair comparison in accuracy (an order of magnitude smaller for the training datasets alone).

In Table 4.23 we present the results for the same BA3MM10 abstraction rules but tested on all 3′ UTR exons and 3′ UTR introns. The models are only trained using random datasets from coding exons and introns due to the lack of sequence samples available for 3′ UTR training data. Little to no intron accuracy loss occurs when one compares the 3′ UTR tested results to CDS-trained/CDS-tested models (0% to -8% net change) while a major loss does occur in the realm of exon accuracy (-17% to -49%). This raises the question of why training on CDS exons works okay for 5′

UTR test data but not on $3'$ UTR test data. The answer is clear: there is a markedly different nucleotide composition between $5'$ and $3'$ UTRs.

Table 4.23: The accuracy of BA3MM10 maps on $3'$ UTR data. Models are trained as usual on the normal training dataset of coding sequences and all introns but tested on the whole dataset of $3'$ UTRs (Table 4.1).

Trained on CDS exons & all introns, tested on $3'$ UTR exons & introns.			
Abstraction Rule / Model	Exon Accuracy	Intron Accuracy	M-value
GC-richness	43%	64%	0.527
GT-richness	44	75	0.563
AG-richness	53	67	0.594
BA3 Best	29	87	0.490
A priori 3	40	69	0.519
SP version 2008	32	77	0.495
SP '08 Optimized (HD4)	30	80	0.485
SP version 2009	28	79	0.468
SP '09 Optimized (HD4)	27	83	0.472
SP '09 Top 24 Positive	27	80	0.462
SP '09 Positive HD4	26	84	0.462
SP '09 Top 24 Negative	34	73	0.498
SP '09 Negative HD4	30	83	0.488

Table 4.24 shows the compositional differences between the datasets (see Table 4.1) of various human genomic regions. These compositional differences are calculated on the level of the 5-mer (overlapping) frequency values for each genomic region. The "difference" between the two frequency distributions used is known as the Jensen-Shannon divergence value, a symmetric function that takes two frequency distributions as input and produces a non-negative real number as a measure of their differentness. Larger values will be more different. A zero value of course indicates the same two frequency distributions. The Jensen-Shannon divergence has been used in the segmentation of genomic regions [34], such as estimating the boundary points between coding and non-coding regions, and so is appropriate for measuring the similitude of different genomic regions.

The first observation one should make is that introns within the coding sequence are very similar to those within the $5'$ or $3'$ UTRs ($D = 0.001$). CDS exons are the

Table 4.24: The Jensen-Shannon divergence (D) matrix for human 5-mer frequency distributions in various genomic regions. A '-' indicates a zero value; in the column heading "EX" is short for exon while "IN" is short for intron; "UTR" stands for untranslated region and "INTER" stands for intergenic region.

$D(F_i, F_j)$	$3'$ EX	$5'$ EX	CDS	INTER	$3'$ IN	$5'$ IN	CDS IN
$3'$ UTR Exon	-	0.231	0.103	0.012	0.008	0.007	0.008
$5'$ UTR Exon	0.231	-	0.097	0.208	0.274	0.256	0.281
CDS Exon	0.103	0.097	-	0.086	0.129	0.125	0.139
Intergenic	0.012	0.208	0.086	-	0.010	0.010	0.014
$3'$ UTR Intron	0.008	0.274	0.129	0.010	-	0.002	0.001
$5'$ UTR Intron	0.007	0.256	0.125	0.010	0.002	-	0.001
CDS Intron	0.008	0.281	0.139	0.014	0.001	0.001	-

most different from introns within the CDS ($D = 0.139$), just as one would expect for proper splicing to occur, and are only a little less different than inrons within the $5'$ and $3'$ UTRs ($D = 0.125$ & 0.129 respectively). CDS exons are, however, quite different from $3'$ UTR exons ($D = 0.103$). This difference unfortunately has a profound impact on detecting $3'$ UTR exons using CDS exon training data. Notice that the Jensen-Shannon Divergence of $3'$ UTR exons have a $D \leq 0.008$ versus any kind of intron. In other words, $3'$ UTR exons are closer to introns in terms of their 5-mer frequency distribution than to CDS exons! Thus, training with CDS exons and all introns and testing with $3'$ UTR exons and introns will produce anti-predictive results ($< 50\%$) for exon accuracy since the model will be trained to see $3'$ UTR exons as introns more often than exons. Clearly, this is just the problem that we have in Table 4.23.

Composition-based approaches may be unsuccessful in predicting $3'$ UTR exons. However, the case is quite different when you look at $5'$ UTR exons. Indeed, $5'$ UTR exons have a $D \geq 0.2$ versus any intron, intergenic region (from this sample), and even versus $3'$ UTR exons. The divergence value ($D = 0.097$) between CDS exons and $5'$ UTR exons is more restrained, although not paltry, which makes good sense given their shared splicing signals but difference in terms of the genetic code. Thus, we conclude that compositional approaches, such as ours, are ripe to identify $5'$ UTR exons for gene finding algorithms, although $5'$ UTR optimized abstraction rules and larger training datasets ought to be used.

In order to help test this assumption we applied our SVM methodology (as before) for model combination and prediction of $5'$ UTR exons, but using coding exons in the training set. We used the BA1 best, BA2 best, BA3 best, and BA4 best abstraction rules as well as the homogeneous nucleotide Markov model of order 6, the duplication model for CDBAMM, and the splicing potential abstraction rule (version 2009) to perform the "choose K out of 7 models" analysis. All BAMM classifiers were of order 10 while the SVM itself used an inhomogeneous, normalized polynomial kernel of degree 3. For a choice of $K = 1$ models, the highest scoring map was the BA3 best abstraction rule, which went from an unoptimized M-value of 0.757 to an SVM optimized M-value of 0.876. The exon and intron accuracy for the optimized BA3 best abstraction rule were 87% and 88% respectively, jumping from unoptimized values of 66% and 93% for exon and intron accuracy. The best combination of any 2 out of 7 models were the BA2 best and BA3 best abstraction rules with an M-value of 0.885, an exon accuracy of approximately 88%, and an intron accuracy of about 90%. Adding more models did not increase prediction accuracy very much. The best combination for the K out of 7 analysis was for the choice of 4 models: the BA1, BA2, and BA3 best abstraction rules along with the splicing potential map. The M-value for this combination of models was 0.888, the exon accuracy was about 87%, and the intron accuracy was approximately 91%. It may be the case that training with coding sequences and testing on untranslated ones limits the increased accuracy gained from the SVM optimization just as it did for the individual unoptimized versions of these classifiers. Nevertheless, this result is much better than a more traditional method—the unoptimized nucleotide, homogeneous Markov model of order 6—that has an M-value of 0.802, an exon accuracy of 87% and an intron accuracy of only 75%. Again we see that our BAMM algorithm shows increased efficacy in intron identification above and beyond the homogeneous nucleotide Markov model approach, suggesting genomic sequence signals within introns may be emphasized by our abstraction rules.

4.4 Final Remarks

Binary-abstracted Markov models can emphasize different biologically relevant genomic sequence patterns—about 10 to 50 bp long—using the conversion of 1-, 2-, 3-, and 4-mer oligonucleotides into binary (0 or 1) sequences. Some of the abstraction rules used to convert the nucleotide sequences to binary ones are frame-independent (when considering coding sequence classification) whereas others are not. Abstraction rules can additionally be stated in terms of nucleotide context, such as for repetitive sequence patterns. The best five individual BAMM classifiers have about the same exon-intron prediction power as the more traditional homogeneous, nucleotide Markov model of order 6. Moreover, after classifier optimization by support vector machine (SVM) technology, two BAMM classifiers exceed the accuracy of the homogeneous Markov model of order 6.

The classifier models built by such abstractions rules can be combined to increase the accuracy in sequence discrimination via machine-learning tools. When combining information from different BAMM classifiers using SVM technology, the six best BAMM combinations achieve a discrimination accuracy of exons and introns at over 95%—making them competitive with other gene-prediction approaches, although admittedly not in terms of producing a sequence parse. In any case, BAMM can also be used not only on coding exons and introns, but also on $5'$ UTR exons and introns. We have shown that SVM optimized BAMM classifiers, such as the BA3 best abstraction rule, can achieve 87% prediction accuracy individually while combinations of 4 classifiers can achieve an average about 89% accuracy. This makes BAMM a powerful tool with great potential for the prediction of 5′ UTR exons.

We believe that given the proper customization of abstraction rules and with correct training data, the BAMM method can be applied to non gene finding problems as well. Rather than to be seen as a replacement for the excellent technologies based on traditional Markov models, we think BAMM is a useful addition to complement and extend what has already be done. In the future we would like to see BAMM applied to more species, integrated into a mature gene finding algorithm with HMM technology (to do sequence parsing, not merely classification), and to see BAMM applied to a more diverse range of sequence classification problems. Moreover, having learned how best to deploy and optimize our initial approach, we foresee being able to next turn our attention, for the immediate future, to the development of BAMM classifiers for alternative splicing recognition.

4.5 Authors' Contributions

SS, GS, AM, and AF wrote and edited the manuscript. SS and GS performed experiments involving machine-learning while SS and AM wrote the main algorithms, optimized the rules, and performed model testing. AF and SS designed the experiments and AF came up with the majority of the ideas for the abstraction rules.

4.6 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0643542. Many thanks to the Ohio Supercomputer Center for their help and answers in the use and operation of Glenn. We are grateful to the Mark Borodovsky lab, GA Tech, for their discussion of concepts related to Markov models and their suggestion of the *a priori* abstraction rule. We express our gratitude to Vadim Filatov, Dinom LLC for his guidance in the conceptual development of context-dependent BAMM maps suitable for exon/intron discrimination. We thank Craig Zirbel, Bowling Green State University, for his advice on the project and the suggestion of using support vector machines. Final thanks to Peter Bazeley, University of Toledo, for his technical support and to Sadik Khuder, University of Toledo, for his discussion on statistical issues.

4.A Additional Tabular Data

Table 4.25: The trial 2 ("half & half" starting seed) optimization of 4-mer abstraction rules using a BA4MM12-like algorithm on the validation set. The optimal value appears in bold.

Seed	Exon Acc.	Intron Acc.	M-value	M-delta	HD
Round 1	65.1%	68.6%	0.6683	0.0000	0
Round 2	66.8	69.7	0.6821	0.0138	3
Round 3	67.4	71.3	0.6929	0.0108	6
Round 4	68.7	72.5	0.7050	0.0121	9
Round 5	69.8	73.1	0.7142	0.0092	12
Round 6	70.4	74.3	0.7231	0.0089	15
Round 7	71.5	75.5	0.7343	0.0112	18
Round 8	72.0	77.0	0.7436	0.0093	21
Round 9	72.7	78.1	0.7524	0.0088	24
Round 10	72.7	79.9	0.7605	0.0081	27
Round 11	72.9	81.8	0.7691	0.0086	30
Round 12	73.4	82.7	0.7755	0.0064	33
Round 13	73.7	83.2	0.7794	0.0039	36
Round 14	74.3	83.5	0.7839	0.0045	39
Round 15	74.5	84.3	0.7881	0.0042	42
Round 16	75.0	84.3	0.7914	0.0033	45
Round 17	75.4	85.0	0.7959	0.0045	48
Round 18	75.5	85.8	0.7999	0.0040	51
Round 19	75.8	86.3	0.8033	0.0034	54
Round 20	75.7	87.5	0.8066	0.0033	57
Round 21	76.1	88.0	0.8107	0.0041	60
Round 22	76.1	88.7	0.8133	0.0026	63
Round 23	76.5	89.0	0.8164	0.0031	66
Round 24	76.4	90.3	0.8195	0.0031	69
Round 25	76.9	90.4	0.8231	0.0036	72
Round 26	77.1	90.4	0.8248	0.0017	75
Round 27	77.4	90.7	0.8274	0.0026	78
Round 28	77.6	90.7	0.8287	0.0013	81
Round 29	77.7	91.5	0.8310	0.0023	84
Round 30	77.8	91.9	0.8327	0.0017	87
Round 31	78.2	91.9	0.8353	0.0026	90
Round 32	78.5	92.2	0.8382	0.0029	93
Round 33	79.0	91.8	0.8407	0.0025	96
Round 34	79.2	92.2	0.8430	0.0023	99
Round 35	79.2	93.0	0.8448	0.0018	102

Continued on next page

Table 4.25: (continued)

Seed	Exon Acc.	Intron Acc.	M-value	M-delta	HD
Round 36	79.4	93.4	0.8467	0.0019	103
Round 37	79.5	93.0	0.8470	0.0003	106
Round 38	79.6	93.1	0.8476	0.0006	107
Round 39	79.6	93.5	0.8483	0.0007	108
Round 40	79.6	93.8	0.8489	0.0006	107
Round 41	79.7	93.5	0.8491	0.0002	106
Round 42	79.7	93.5	0.8493	0.0002	107
Round 43	79.6	93.8	0.8494	0.0001	107
Round 44	79.5	94.1	0.8494	0.0000	108
Round 45	**79.5**	**94.3**	**0.8495**	**0.0001**	**107**
Round 46	79.5	94.1	0.8494	-0.0001	108

Table 4.26: The trial 3 ("striped" starting seed) optimization of 4-mer abstraction rules using a BA4MM12-like algorithm on the validation set. The optimal value appears in bold.

Seed	Exon Acc.	Intron Acc.	M-value	M-delta	HD
Round 1	62.3%	76.6%	0.6865	0.0000	0
Round 2	65.2	80.1	0.7161	0.0296	3
Round 3	67.2	82.0	0.7353	0.0192	6
Round 4	68.2	84.4	0.7500	0.0147	9
Round 5	68.8	86.4	0.7595	0.0095	12
Round 6	69.9	86.3	0.7662	0.0067	15
Round 7	70.5	87.0	0.7721	0.0059	18
Round 8	71.2	87.9	0.7789	0.0068	21
Round 9	72.2	87.2	0.7836	0.0047	24
Round 10	72.8	87.9	0.7897	0.0061	27
Round 11	73.3	88.7	0.7950	0.0053	30
Round 12	73.9	88.6	0.7985	0.0035	33
Round 13	74.4	88.8	0.8026	0.0041	36
Round 14	74.8	89.5	0.8071	0.0045	39
Round 15	75.2	90.2	0.8111	0.0040	42
Round 16	75.5	90.2	0.8133	0.0022	45
Round 17	75.9	90.2	0.8157	0.0024	48
Round 18	76.0	90.6	0.8174	0.0017	51
Round 19	76.4	90.3	0.8195	0.0021	54
Round 20	76.7	90.3	0.8218	0.0023	57
Round 21	77.2	90.0	0.8241	0.0023	60
Round 22	77.4	91.1	0.8283	0.0042	63
Round 23	77.7	91.8	0.8316	0.0033	66

Continued on next page

Table 4.26: (continued)

Seed	Exon Acc.	Intron Acc.	M-value	M-delta	HD
Round 24	77.9	92.0	0.8336	0.0020	67
Round 25	78.2	91.8	0.8352	0.0016	70
Round 26	78.5	91.5	0.8365	0.0013	73
Round 27	78.7	92.3	0.8400	0.0035	76
Round 28	78.9	93.2	0.8432	0.0032	79
Round 29	78.9	93.2	0.8436	0.0004	82
Round 30	78.9	93.9	0.8445	0.0009	83
Round 31	79.2	94.0	0.8469	0.0024	86
Round 32	79.4	93.5	0.8472	0.0003	87
Round 33	79.7	93.1	0.8484	0.0012	88
Round 34	79.7	93.4	0.8489	0.0005	90
Round 35	79.8	93.5	0.8498	0.0009	93
Round 36	79.8	93.8	0.8503	0.0005	94
Round 37	80.0	93.2	0.8504	0.0001	97
Round 38	79.8	94.0	0.8513	0.0009	98
Round 39	**80.0**	**93.9**	**0.8519**	**0.0006**	**101**
Round 40	79.9	94.0	0.8519	0.0000	103
Round 41	80.0	93.9	0.8519	0.0000	101

Table 4.27: Abstraction rules for the three best BA4 optimization trials. A comparison to BA2 best is also given by dividing each tetranucleotide into a left and right dinucleotide. The consensus or agreement of the BA4 optimization trials is given under the "Con." field while the consensus of the trials plus the BA2 best map dinucleotides is given under the "All" field in fractional form.

4-mer	T1	T2	T2	Con.	L 2-mer	BA2	R 2-mer	BA2	All
AAAA	0	0	0	3/3	AA	1	AA	1	3/4
AAAC	1	0	1	2/3	AA	1	AC	1	4/5
AAAG	1	1	1	3/3	AA	1	AG	1	5/5
AAAT	0	0	0	3/3	AA	1	AT	1	3/5
AACA	1	1	1	3/3	AA	1	CA	1	5/5
AACC	1	1	0	2/3	AA	1	CC	1	4/5
AACG	1	1	1	3/3	AA	1	CG	1	5/5
AACT	1	1	1	3/3	AA	1	CT	1	5/5
AAGA	1	1	1	3/3	AA	1	GA	1	5/5
AAGC	1	1	1	3/3	AA	1	GC	1	5/5
AAGG	1	1	0	2/3	AA	1	GG	0	3/5
AAGT	1	1	1	3/3	AA	1	GT	0	4/5
AATA	0	0	0	3/3	AA	1	TA	0	4/5

Continued on next page

Table 4.27: (continued)

4-mer	T1	T2	T2	Con.	L 2-mer	BA2	R 2-mer	BA2	All
AATC	1	1	1	3/3	AA	1	TC	0	4/5
AATG	1	1	1	3/3	AA	1	TG	1	5/5
AATT	0	0	0	3/3	AA	1	TT	0	4/5
ACAA	1	1	1	3/3	AC	1	AA	1	5/5
ACAC	1	0	1	2/3	AC	1	AC	1	3/4
ACAG	1	1	1	3/3	AC	1	AG	1	5/5
ACAT	1	0	1	2/3	AC	1	AT	1	4/5
ACCA	1	1	1	3/3	AC	1	CA	1	5/5
ACCC	1	0	1	2/3	AC	1	CC	1	4/5
ACCG	1	1	1	3/3	AC	1	CG	1	5/5
ACCT	1	1	1	3/3	AC	1	CT	1	5/5
ACGA	1	1	1	3/3	AC	1	GA	1	5/5
ACGC	0	1	1	2/3	AC	1	GC	1	4/5
ACGG	1	1	1	3/3	AC	1	GG	0	4/5
ACGT	1	1	1	3/3	AC	1	GT	0	4/5
ACTA	1	1	0	2/3	AC	1	TA	0	3/5
ACTC	0	0	1	2/3	AC	1	TC	0	3/5
ACTG	1	1	1	3/3	AC	1	TG	1	5/5
ACTT	0	0	0	3/3	AC	1	TT	0	4/5
AGAA	1	1	1	3/3	AG	1	AA	1	5/5
AGAC	1	1	1	3/3	AG	1	AC	1	5/5
AGAG	1	0	0	2/3	AG	1	AG	1	2/4
AGAT	1	1	1	3/3	AG	1	AT	1	5/5
AGCA	1	1	1	3/3	AG	1	CA	1	5/5
AGCC	0	0	0	3/3	AG	1	CC	1	3/5
AGCG	0	1	0	2/3	AG	1	CG	1	3/5
AGCT	1	0	0	2/3	AG	1	CT	1	3/5
AGGA	1	1	1	3/3	AG	1	GA	1	5/5
AGGC	0	0	0	3/3	AG	1	GC	1	3/5
AGGG	0	0	0	3/3	AG	1	GG	0	4/5
AGGT	0	0	0	3/3	AG	1	GT	0	4/5
AGTA	0	0	0	3/3	AG	1	TA	0	4/5
AGTC	1	0	0	2/3	AG	1	TC	0	3/5
AGTG	0	0	0	3/3	AG	1	TG	1	3/5
AGTT	0	0	0	3/3	AG	1	TT	0	4/5
ATAA	0	0	0	3/3	AT	1	AA	1	3/5
ATAC	0	0	0	3/3	AT	1	AC	1	3/5
ATAG	0	0	0	3/3	AT	1	AG	1	3/5
ATAT	0	0	0	3/3	AT	1	AT	1	3/4
ATCA	1	1	1	3/3	AT	1	CA	1	5/5
ATCC	1	1	1	3/3	AT	1	CC	1	5/5

Continued on next page

Table 4.27: (continued)

4-mer	T1	T2	T2	Con.	L 2-mer	BA2	R 2-mer	BA2	All
ATCG	1	1	1	3/3	AT	1	CG	1	5/5
ATCT	1	1	1	3/3	AT	1	CT	1	5/5
ATGA	1	1	1	3/3	AT	1	GA	1	5/5
ATGC	1	1	1	3/3	AT	1	GC	1	5/5
ATGG	1	1	1	3/3	AT	1	GG	0	4/5
ATGT	1	0	1	2/3	AT	1	GT	0	3/5
ATTA	0	0	0	3/3	AT	1	TA	0	4/5
ATTC	0	0	0	3/3	AT	1	TC	0	4/5
ATTG	1	0	1	2/3	AT	1	TG	1	4/5
ATTT	0	0	0	3/3	AT	1	TT	0	4/5
CAAA	1	1	1	3/3	CA	1	AA	1	5/5
CAAC	1	1	1	3/3	CA	1	AC	1	5/5
CAAG	1	1	1	3/3	CA	1	AG	1	5/5
CAAT	1	1	1	3/3	CA	1	AT	1	5/5
CACA	1	0	0	2/3	CA	1	CA	1	2/4
CACC	0	1	1	2/3	CA	1	CC	1	4/5
CACG	1	1	0	2/3	CA	1	CG	1	4/5
CACT	1	1	1	3/3	CA	1	CT	1	5/5
CAGA	1	1	1	3/3	CA	1	GA	1	5/5
CAGC	1	1	1	3/3	CA	1	GC	1	5/5
CAGG	0	0	0	3/3	CA	1	GG	0	4/5
CAGT	1	1	1	3/3	CA	1	GT	0	4/5
CATA	0	0	0	3/3	CA	1	TA	0	4/5
CATC	1	1	1	3/3	CA	1	TC	0	4/5
CATG	1	1	1	3/3	CA	1	TG	1	5/5
CATT	1	0	0	2/3	CA	1	TT	0	3/5
CCAA	1	1	1	3/3	CC	1	AA	1	5/5
CCAC	1	1	0	2/3	CC	1	AC	1	4/5
CCAG	1	1	1	3/3	CC	1	AG	1	5/5
CCAT	1	1	1	3/3	CC	1	AT	1	5/5
CCCA	0	0	0	3/3	CC	1	CA	1	3/5
CCCC	0	0	0	3/3	CC	1	CC	1	3/4
CCCG	1	0	0	2/3	CC	1	CG	1	3/5
CCCT	0	0	0	3/3	CC	1	CT	1	3/5
CCGA	1	1	1	3/3	CC	1	GA	1	5/5
CCGC	1	1	1	3/3	CC	1	GC	1	5/5
CCGG	1	1	1	3/3	CC	1	GG	0	4/5
CCGT	1	1	1	3/3	CC	1	GT	0	4/5
CCTA	0	1	0	2/3	CC	1	TA	0	3/5
CCTC	0	0	0	3/3	CC	1	TC	0	4/5
CCTG	0	0	0	3/3	CC	1	TG	1	3/5

Continued on next page

Table 4.27: (continued)

4-mer	T1	T2	T2	Con.	L 2-mer	BA2	R 2-mer	BA2	All
CCTT	0	0	0	3/3	CC	1	TT	0	4/5
CGAA	1	1	1	3/3	CG	1	AA	1	5/5
CGAC	1	1	1	3/3	CG	1	AC	1	5/5
CGAG	1	1	1	3/3	CG	1	AG	1	5/5
CGAT	1	1	1	3/3	CG	1	AT	1	5/5
CGCA	1	1	0	2/3	CG	1	CA	1	4/5
CGCC	1	0	1	2/3	CG	1	CC	1	4/5
CGCG	0	1	0	2/3	CG	1	CG	1	2/4
CGCT	1	1	1	3/3	CG	1	CT	1	5/5
CGGA	1	1	1	3/3	CG	1	GA	1	5/5
CGGC	1	1	1	3/3	CG	1	GC	1	5/5
CGGG	0	0	0	3/3	CG	1	GG	0	4/5
CGGT	1	0	0	2/3	CG	1	GT	0	3/5
CGTA	1	1	1	3/3	CG	1	TA	0	4/5
CGTC	1	1	1	3/3	CG	1	TC	0	4/5
CGTG	1	0	1	2/3	CG	1	TG	1	4/5
CGTT	1	1	0	2/3	CG	1	TT	0	3/5
CTAA	0	0	0	3/3	CT	1	AA	1	3/5
CTAC	1	1	1	3/3	CT	1	AC	1	5/5
CTAG	0	0	0	3/3	CT	1	AG	1	3/5
CTAT	1	0	0	2/3	CT	1	AT	1	3/5
CTCA	1	0	0	2/3	CT	1	CA	1	3/5
CTCC	1	0	1	2/3	CT	1	CC	1	4/5
CTCG	1	1	0	2/3	CT	1	CG	1	4/5
CTCT	0	0	0	3/3	CT	1	CT	1	3/4
CTGA	1	1	1	3/3	CT	1	GA	1	5/5
CTGC	1	1	1	3/3	CT	1	GC	1	5/5
CTGG	1	1	1	3/3	CT	1	GG	0	4/5
CTGT	1	1	0	2/3	CT	1	GT	0	3/5
CTTA	0	0	0	3/3	CT	1	TA	0	4/5
CTTC	1	1	1	3/3	CT	1	TC	0	4/5
CTTG	1	0	0	2/3	CT	1	TG	1	3/5
CTTT	0	0	0	3/3	CT	1	TT	0	4/5
GAAA	1	1	1	3/3	GA	1	AA	1	5/5
GAAC	1	1	1	3/3	GA	1	AC	1	5/5
GAAG	1	1	1	3/3	GA	1	AG	1	5/5
GAAT	1	1	1	3/3	GA	1	AT	1	5/5
GACA	1	1	1	3/3	GA	1	CA	1	5/5
GACC	1	1	1	3/3	GA	1	CC	1	5/5
GACG	1	1	1	3/3	GA	1	CG	1	5/5
GACT	1	1	1	3/3	GA	1	CT	1	5/5

Continued on next page

Table 4.27: (continued)

4-mer	T1	T2	T2	Con.	L 2-mer	BA2	R 2-mer	BA2	All
GAGA	1	1	1	3/3	GA	1	GA	1	4/4
GAGC	1	1	1	3/3	GA	1	GC	1	5/5
GAGG	1	0	0	2/3	GA	1	GG	0	3/5
GAGT	1	1	0	2/3	GA	1	GT	0	3/5
GATA	1	1	1	3/3	GA	1	TA	0	4/5
GATC	1	1	1	3/3	GA	1	TC	0	4/5
GATG	1	1	1	3/3	GA	1	TG	1	5/5
GATT	1	1	1	3/3	GA	1	TT	0	4/5
GCAA	1	1	1	3/3	GC	1	AA	1	5/5
GCAC	1	1	0	2/3	GC	1	AC	1	4/5
GCAG	1	1	1	3/3	GC	1	AG	1	5/5
GCAT	1	1	0	2/3	GC	1	AT	1	4/5
GCCA	0	1	1	2/3	GC	1	CA	1	4/5
GCCC	1	0	1	2/3	GC	1	CC	1	4/5
GCCG	1	1	1	3/3	GC	1	CG	1	5/5
GCCT	0	0	0	3/3	GC	1	CT	1	3/5
GCGA	1	1	1	3/3	GC	1	GA	1	5/5
GCGC	0	0	1	2/3	GC	1	GC	1	2/4
GCGG	1	0	1	2/3	GC	1	GG	0	3/5
GCGT	0	1	1	2/3	GC	1	GT	0	3/5
GCTA	1	1	1	3/3	GC	1	TA	0	4/5
GCTC	1	1	1	3/3	GC	1	TC	0	4/5
GCTG	1	1	1	3/3	GC	1	TG	1	5/5
GCTT	0	1	0	2/3	GC	1	TT	0	3/5
GGAA	1	1	1	3/3	GG	0	AA	1	4/5
GGAC	1	1	1	3/3	GG	0	AC	1	4/5
GGAG	0	1	1	2/3	GG	0	AG	1	3/5
GGAT	1	1	1	3/3	GG	0	AT	1	4/5
GGCA	1	0	1	2/3	GG	0	CA	1	3/5
GGCC	0	0	1	2/3	GG	0	CC	1	3/5
GGCG	0	0	0	3/3	GG	0	CG	1	4/5
GGCT	0	0	0	3/3	GG	0	CT	1	4/5
GGGA	0	0	0	3/3	GG	0	GA	1	4/5
GGGC	0	0	0	3/3	GG	0	GC	1	4/5
GGGG	0	0	0	3/3	GG	0	GG	0	4/4
GGGT	0	0	0	3/3	GG	0	GT	0	5/5
GGTA	1	0	0	2/3	GG	0	TA	0	4/5
GGTC	1	0	1	2/3	GG	0	TC	0	3/5
GGTG	0	0	0	3/3	GG	0	TG	1	4/5
GGTT	1	0	0	2/3	GG	0	TT	0	4/5
GTAA	0	0	0	3/3	GT	0	AA	1	4/5

Continued on next page

Table 4.27: (continued)

4-mer	T1	T2	T2	Con.	L 2-mer	BA2	R 2-mer	BA2	All
GTAC	1	1	1	$^3/_3$	GT	0	AC	1	$^4/_5$
GTAG	0	0	0	$^3/_3$	GT	0	AG	1	$^4/_5$
GTAT	0	0	0	$^3/_3$	GT	0	AT	1	$^4/_5$
GTCA	1	0	1	$^2/_3$	GT	0	CA	1	$^3/_5$
GTCC	1	0	1	$^2/_3$	GT	0	CC	1	$^3/_5$
GTCG	1	1	1	$^3/_3$	GT	0	CG	1	$^4/_5$
GTCT	0	0	0	$^3/_3$	GT	0	CT	1	$^4/_5$
GTGA	1	1	1	$^3/_3$	GT	0	GA	1	$^4/_5$
GTGC	1	1	1	$^3/_3$	GT	0	GC	1	$^4/_5$
GTGG	1	0	0	$^2/_3$	GT	0	GG	0	$^4/_5$
GTGT	1	0	0	$^2/_3$	GT	0	GT	0	$^3/_4$
GTTA	0	0	0	$^3/_3$	GT	0	TA	0	$^5/_5$
GTTC	1	0	1	$^2/_3$	GT	0	TC	0	$^3/_5$
GTTG	0	0	1	$^2/_3$	GT	0	TG	1	$^3/_5$
GTTT	0	0	0	$^3/_3$	GT	0	TT	0	$^5/_5$
TAAA	0	0	0	$^3/_3$	TA	0	AA	1	$^4/_5$
TAAC	0	0	0	$^3/_3$	TA	0	AC	1	$^4/_5$
TAAG	0	0	0	$^3/_3$	TA	0	AG	1	$^4/_5$
TAAT	0	0	0	$^3/_3$	TA	0	AT	1	$^4/_5$
TACA	1	1	1	$^3/_3$	TA	0	CA	1	$^4/_5$
TACC	1	1	1	$^3/_3$	TA	0	CC	1	$^4/_5$
TACG	1	1	1	$^3/_3$	TA	0	CG	1	$^4/_5$
TACT	0	0	0	$^3/_3$	TA	0	CT	1	$^4/_5$
TAGA	0	0	0	$^3/_3$	TA	0	GA	1	$^4/_5$
TAGC	0	0	0	$^3/_3$	TA	0	GC	1	$^4/_5$
TAGG	0	0	0	$^3/_3$	TA	0	GG	0	$^5/_5$
TAGT	0	0	0	$^3/_3$	TA	0	GT	0	$^5/_5$
TATA	0	0	0	$^3/_3$	TA	0	TA	0	$^4/_4$
TATC	1	0	0	$^2/_3$	TA	0	TC	0	$^4/_5$
TATG	0	0	0	$^3/_3$	TA	0	TG	1	$^4/_5$
TATT	0	0	0	$^3/_3$	TA	0	TT	0	$^5/_5$
TCAA	1	1	1	$^3/_3$	TC	0	AA	1	$^4/_5$
TCAC	1	0	1	$^2/_3$	TC	0	AC	1	$^3/_5$
TCAG	1	1	1	$^3/_3$	TC	0	AG	1	$^4/_5$
TCAT	1	0	1	$^2/_3$	TC	0	AT	1	$^3/_5$
TCCA	1	1	1	$^3/_3$	TC	0	CA	1	$^4/_5$
TCCC	0	0	0	$^3/_3$	TC	0	CC	1	$^4/_5$
TCCG	1	1	1	$^3/_3$	TC	0	CG	1	$^4/_5$
TCCT	0	0	0	$^3/_3$	TC	0	CT	1	$^4/_5$
TCGA	1	1	1	$^3/_3$	TC	0	GA	1	$^4/_5$
TCGC	1	1	1	$^3/_3$	TC	0	GC	1	$^4/_5$

Continued on next page

Table 4.27: (continued)

4-mer	T1	T2	T2	Con.	L 2-mer	BA2	R 2-mer	BA2	All
TCGG	1	1	1	3/3	TC	0	GG	0	3/5
TCGT	1	0	1	2/3	TC	0	GT	0	3/5
TCTA	0	0	0	3/3	TC	0	TA	0	5/5
TCTC	0	0	0	3/3	TC	0	TC	0	4/4
TCTG	0	0	1	2/3	TC	0	TG	1	3/5
TCTT	0	0	0	3/3	TC	0	TT	0	5/5
TGAA	1	1	1	3/3	TG	1	AA	1	5/5
TGAC	1	1	1	3/3	TG	1	AC	1	5/5
TGAG	0	0	0	3/3	TG	1	AG	1	3/5
TGAT	1	0	1	2/3	TG	1	AT	1	4/5
TGCA	1	1	1	3/3	TG	1	CA	1	5/5
TGCC	1	1	1	3/3	TG	1	CC	1	5/5
TGCG	1	0	1	2/3	TG	1	CG	1	4/5
TGCT	1	1	1	3/3	TG	1	CT	1	5/5
TGGA	1	1	1	3/3	TG	1	GA	1	5/5
TGGC	1	1	1	3/3	TG	1	GC	1	5/5
TGGG	0	0	0	3/3	TG	1	GG	0	4/5
TGGT	1	0	0	2/3	TG	1	GT	0	3/5
TGTA	0	0	0	3/3	TG	1	TA	0	4/5
TGTC	0	0	1	2/3	TG	1	TC	0	3/5
TGTG	1	0	0	2/3	TG	1	TG	1	2/4
TGTT	0	0	0	3/3	TG	1	TT	0	4/5
TTAA	0	0	0	3/3	TT	0	AA	1	4/5
TTAC	0	0	1	2/3	TT	0	AC	1	3/5
TTAG	0	0	0	3/3	TT	0	AG	1	4/5
TTAT	0	0	0	3/3	TT	0	AT	1	4/5
TTCA	1	0	1	2/3	TT	0	CA	1	3/5
TTCC	0	0	1	2/3	TT	0	CC	1	3/5
TTCG	1	1	1	3/3	TT	0	CG	1	4/5
TTCT	0	0	0	3/3	TT	0	CT	1	4/5
TTGA	0	0	1	2/3	TT	0	GA	1	3/5
TTGC	1	0	1	2/3	TT	0	GC	1	3/5
TTGG	1	0	1	2/3	TT	0	GG	0	3/5
TTGT	0	0	0	3/3	TT	0	GT	0	5/5
TTTA	0	0	0	3/3	TT	0	TA	0	5/5
TTTC	0	0	0	3/3	TT	0	TC	0	5/5
TTTG	0	0	0	3/3	TT	0	TG	1	4/5
TTTT	0	0	0	3/3	TT	0	TT	0	4/4

4.B Source Code

4.B.1 BAMM: convertTrainTestMM.pl

```
#!/usr/bin/perl
# convertTrainTestMM.pl - Samuel Shepard - 11.2009
# Convert a nucleotide sequence to binary based on the map.
# Train and test the sequences using a binary markov model.
# This program is memory intensive.

# Define some global variables, necessary for enumerate().
$K       = 0;
%map     = ();
@letters = ( ’a’, ’t’, ’c’, ’g’ );

sub enumerate($$) {
  my $level = shift;
  my $tail  = shift;
  my $base;
  if ( $level == 0 ) {
    $map{$tail} = $K;
    $K++;
  } else {
    for $base (@letters) {
      enumerate( ( $level - 1 ), "$tail$base" );
    }
  }
}

# GET the input parameters.
if ( scalar(@ARGV) < 5 ) {
  $message = "\nUsage:\n\tperl $0 ";
  $message .= "\n\t\t <sample_directory> <output_file> <map_file>";
  $message .= "\n\t\t <HMMorder> <MAPorder/abstraction level> [frame]\n";
  die($message);
}

# PROCESS the parameters.
$mmOrder  = $ARGV[3];    # The markov model order.
$mapOrder = $ARGV[4];    # The map order/abstraction level.

# Frame shift the data if told to.
if ( scalar(@ARGV) > 5 ) {
  $frameshift = $ARGV[5];
} else {
  $frameshift = 0;
}

# Store the abstraction rules.
open( MAPS, '<', $ARGV[2] ) or die("Could not open map file $ARGV[2].\n");
$/    = "\n";
@maps = <MAPS>;
chomp(@maps);
$number_maps = scalar(@maps);
close(MAPS);

# Read in the datasets.
# Currently assumes fixed naming scheme of exons v. introns.
open( ITRAIN, '<', "$ARGV[0]/training_intron.fa" )
  or die("Cannot open $ARGV[0]/training_intron.fa\n");
open( ETRAIN, '<', "$ARGV[0]/training_exon.fa" )
  or die("Cannot open $ARGV[0]/training_exon.fa\n");
open( ITEST, '<', "$ARGV[0]/test_intron.fa" )
  or die("Cannot open $ARGV[0]/test_intron.fa\n");
open( ETEST, '<', "$ARGV[0]/test_exon.fa" )
  or die("Cannot open $ARGV[0]/test_exon.fa\n");

# INITIALIZE variables.
@handles      = ( 'ITRAIN', 'ETRAIN', 'ITEST', 'ETEST' );
$/            = '>';
$i            = 0;
$itrain_start = $itrain_end = 0;
$etrain_start = $etrain_end = 0;
$itest_start  = $itest_end = 0;
$etest_start  = $etest_end = 0;

$initBits         = $mmOrder;
$transBits        = $mmOrder + 1;
$minimum_length   = $mapOrder * ( $mmOrder + 1 );
$remaining_length = 0;

# READ sequence files.
foreach $handle (@handles) {

  if ( $handle eq 'ITRAIN' ) {
    $itrain_start = $i;
  } elsif ( $handle eq 'ETRAIN' ) {
    $etrain_start = $i;
  } elsif ( $handle eq 'ITEST' ) {
    $itest_start = $i;
  } else {
    $etest_start = $i;
  }

  while ( $record = <$handle> ) {
    @lines = split( "\n", $record );

    $header = shift(@lines);
    chomp(@lines);
    $sequence = lc( join( '', @lines ) );

    $current_length = length($sequence);

    # if too short, skip it.
    $remaining_length = $current_length - $frameshift;
    if ( $remaining_length < $minimum_length ) {
      next;
    } else {
      $nt_count = ( $sequence =~ tr/atgc// );
      if ( $current_length != $nt_count ) {
        next;
      } else {
        $sequences[$i] = substr( $sequence, $frameshift, $remaining_length );
        $i++;
      }
    }
  }

  if ( $handle eq 'ITRAIN' ) {
    $itrain_end = $i;
  } elsif ( $handle eq 'ETRAIN' ) {
    $etrain_end = $i;
  } elsif ( $handle eq 'ITEST' ) {
    $itest_end = $i;
  } else {
    $etest_end = $i;
  }

  close($handle);
}

# PROCESS abstraction rules.
# Map order (e.g., the abstraction level).
if ( $mapOrder == 3 ) {

  # load map
  $map{'ttc'} = 0;
  $map{'tgt'} = 1;
  $map{'ctt'} = 2;
```

```
$map{'att'} = 3;
$map{'agg'} = 4;
$map{'atg'} = 5;
$map{'gcg'} = 6;
$map{'tta'} = 7;
$map{'gct'} = 8;
$map{'caa'} = 9;
$map{'aat'} = 10;
$map{'acg'} = 11;
$map{'cgt'} = 12;
$map{'tac'} = 13;
$map{'cta'} = 14;
$map{'cga'} = 15;
$map{'aca'} = 16;
$map{'ggc'} = 17;
$map{'tgg'} = 18;
$map{'ccg'} = 19;
$map{'gca'} = 20;
$map{'ggt'} = 21;
$map{'tcg'} = 22;
$map{'acc'} = 23;
$map{'cat'} = 24;
$map{'gag'} = 25;
$map{'gtc'} = 26;
$map{'act'} = 27;
$map{'tcc'} = 28;
$map{'cct'} = 29;
$map{'gtg'} = 30;
$map{'ttg'} = 31;
$map{'agc'} = 32;
$map{'atc'} = 33;
$map{'gaa'} = 34;
$map{'cca'} = 35;
$map{'gat'} = 36;
$map{'ttt'} = 37;
$map{'ctg'} = 38;
$map{'cgc'} = 39;
$map{'aac'} = 40;
$map{'tag'} = 41;
$map{'tct'} = 42;
$map{'gta'} = 43;
$map{'tgc'} = 44;
$map{'gac'} = 45;
$map{'taa'} = 46;
$map{'ccc'} = 47;
$map{'ggg'} = 48;
$map{'cag'} = 49;
```

```
  $map{'aaa'} = 50;
  $map{'aag'} = 51;
  $map{'ctc'} = 52;
  $map{'aga'} = 53;
  $map{'tca'} = 54;
  $map{'tga'} = 55;
  $map{'cgg'} = 56;
  $map{'ata'} = 57;
  $map{'gtt'} = 58;
  $map{'gga'} = 59;
  $map{'gcc'} = 60;
  $map{'cac'} = 61;
  $map{'agt'} = 62;
  $map{'tat'} = 63;
} else {
  enumerate( $mapOrder, '' );
}

# INITIALIZE markov model variables.
%initE = %initI = %transE = %transI = ();
$patternsInit  = 2**$initBits;
$patternsTrans = 2**$transBits;

# Open output file.
open( OUT, '>', "$ARGV[1]" ) or die("Cannot open $ARGV[1].\n");

# PROCESS each map consecutively
$mapI          = 0;
$map_size      = 4**$mapOrder;
@bit_sequences = @scoresI = @scoresE = ();
for ( $mapI = 0 ; $mapI < $number_maps ; $mapI++ ) {

  # Initialize probabilities.
  $template = '%0' . $initBits . 'b';
  for ( $i = 0 ; $i < $patternsInit ; $i++ ) {
    $key = sprintf( $template, $i );
    $initE{$key} = $initI{$key} = 0;
  }

  $template = '%0' . $transBits . 'b';
  for ( $i = 0 ; $i < $patternsTrans ; $i++ ) {
    $key = sprintf( $template, $i );
    $transE{$key} = $transI{$key} = 0;
  }
  $sumInitI = $sumInitE = $sumTransI = $sumTransE = 0;

  # Load the abstraction rule.
```

```
if ( length( $maps[$mapI] ) != ($map_size) ) {
  die("Map '$maps[$mapI]' not correct length.\n");
}
@bits = split( '', $maps[$mapI] );
if ( $map_size != scalar(@bits) ) { die("Map size mismatch.\n"); }

# Convert each sequence according to the current abstraction rule.
for ( $i = 0 ; $i < $etest_end ; $i++ ) {
  $current_length = length( $sequences[$i] );
  $bit_sequences[$i] = '';
  for (
    $pos = 0 ;
    $pos <= ( $current_length - $mapOrder ) ;
    $pos += $mapOrder
    )
  {
    $key = substr( $sequences[$i], $pos, $mapOrder );
    $bit_sequences[$i] .= $bits[ $map{$key} ];
  }
}

# TRAIN the BAMM classifier.
# Analyze the intron sequences.
$totalI = 0;
for ( $i = $itrain_start ; $i < $itrain_end ; $i++ ) {
  $current_length = length( $bit_sequences[$i] );
  for ( $pos = 0 ; $pos <= ( $current_length - $initBits ) ; $pos++ ) {
    $key = substr( $bit_sequences[$i], $pos, $initBits );
    $initI{$key}++;
    $sumInitI++;
  }

  for ( $pos = 0 ; $pos <= ( $current_length - $transBits ) ; $pos++ ) {
    $key = substr( $bit_sequences[$i], $pos, $transBits );
    $transI{$key}++;
    $sumTransI++;
  }
}

# Analyze the exon sequences.
$totalE = 0;
for ( $i = $etrain_start ; $i < $etrain_end ; $i++ ) {
  $current_length = length( $bit_sequences[$i] );
  for ( $pos = 0 ; $pos <= ( $current_length - $initBits ) ; $pos++ ) {
    $key = substr( $bit_sequences[$i], $pos, $initBits );
    $initE{$key}++;
    $sumInitE++;
  }

  for ( $pos = 0 ; $pos <= ( $current_length - $transBits ) ; $pos++ ) {
    $key = substr( $bit_sequences[$i], $pos, $transBits );
    $transE{$key}++;
    $sumTransE++;
  }
}

# Convert occurrences to frequencies.
foreach $key ( keys(%initE) ) {
  $initE{$key} /= $sumInitE;
  $initI{$key} /= $sumInitI;
}

$emptyProb = 0;
$totalUses = 0;

# Test the introns group.
for ( $i = $itest_start ; $i < $itest_end ; $i++ ) {
  $current_length  = length( $bit_sequences[$i] );
  $which           = $i - $itest_start;
  $scoresI[$which] = 0;

  $key = substr( $bit_sequences[$i], 0, $initBits );
  if ( $initI{$key} != 0 ) {
    $probI = log( $initI{$key} );
    $totalUses++;
  } else {
    $emptyProb++;
  }

  if ( $initE{$key} ) {
    $probE = log( $initE{$key} );
    $totalUses++;
  } else {
    $emptyProb++;
  }

  for ( $pos = 0 ; $pos <= ( $current_length - $transBits ) ; $pos++ ) {

    $key  = substr( $bit_sequences[$i], $pos, $transBits );
    $key0 = substr( $key,               0,    $initBits ) . '0';
    $key1 = substr( $key,               0,    $initBits ) . '1';

    if ( $transI{$key0} != 0 && $transI{$key1} != 0 ) {
      $totalUses++;
```

```
        $probI += log( $transI{$key} / ( $transI{$key0} + $transI{$key1} ) );
      } else {
        $emptyProb++;
      }
      if ( $transE{$key0} != 0 && $transE{$key1} != 0 ) {
        $totalUses++;
        $probE += log( $transE{$key} / ( $transE{$key0} + $transE{$key1} ) );
      } else {
        $emptyProb++;
      }
    }

    $scoresI[$which] = $probE - $probI;
  }

  # Test the exons group.
  for ( $i = $etest_start ; $i < $etest_end ; $i++ ) {
    $current_length  = length( $bit_sequences[$i] );
    $which           = $i - $etest_start;
    $scoresE[$which] = 0;

    $key = substr( $bit_sequences[$i], 0, $initBits );

    if ( $initI{$key} != 0 ) {
      $totalUses++;
      $probI = log( $initI{$key} );
    } else {
      $emptyProb++;
    }

    if ( $initE{$key} ) {
      $totalUses++;
      $probE = log( $initE{$key} );
    } else {
      $emptyProb++;
    }

    for ( $pos = 0 ; $pos <= ( $current_length - $transBits ) ; $pos++ ) {

      $key  = substr( $bit_sequences[$i], $pos, $transBits );
      $key0 = substr( $key,               0,    $initBits ) . '0';
      $key1 = substr( $key,               0,    $initBits ) . '1';
      if ( $transI{$key0} != 0 && $transI{$key1} != 0 ) {
        $totalUses++;
        $probI += log( $transI{$key} / ( $transI{$key0} + $transI{$key1} ) );
      } else {
        $emptyProb++;
      }
      if ( $transE{$key0} != 0 && $transE{$key1} != 0 ) {
        $totalUses++;
        $probE += log( $transE{$key} / ( $transE{$key0} + $transE{$key1} ) );
      } else {
        $emptyProb++;
      }
    }
    $scoresE[$which] = $probE - $probI;
  }

  # Output the result (classification scores).
  # Print out the statistics for the particular abstraction rule used.
  $exponent     = log( $totalUses + $emptyProb ) / log(10);
  $possibleUses = $emptyProb + $totalUses;
  $knowledgeUsage =
    sprintf("%0.3f", (($totalUses)**$exponent / $possibleUses**$exponent));
  print "$knowledgeUsage\t($emptyProb)\t$maps[$mapI].\n";
  print OUT $maps[$mapI], "\n", join( ' ', @scoresI ), "\n",
    join( ' ', @scoresE ), "\n\n";
}
close(OUT);
```

4.B.2 Homogeneous Markov model: mcClassifier.pl

```
#!/usr/bin/perl
# mcClassifier.pl - Sam Shepard - 11.2009
# Homogeneous markov chain classifier.

# GET the input parameters.
if ( scalar(@ARGV) < 3 ) {
  $message = "\nUsage:\n\tperl $0 <sample_directory>";
  $message .= " <output_file> <MMorder> [frame]\n";
  die($message);
}

# PROCESS parameters
if ( scalar(@ARGV) > 3 ) {
  $frameshift = $ARGV[3];
} else {
  $frameshift = 0;
}

# Trim function.
# Removes whitespace from the start and end of the string
```

```
sub trim($) {
  my $string = shift;
  $string =~ /^\s*(.*?)\s*$/;
  return $1;
}

# read samples
open( ITRAIN, '<', "$ARGV[0]/training_intron.fa" )
  or die("Cannot open $ARGV[0]/training_intron.fa\n");
open( ETRAIN, '<', "$ARGV[0]/training_exon.fa" )
  or die("Cannot open $ARGV[0]/training_exon.fa\n");
open( ITEST, '<', "$ARGV[0]/test_intron.fa" )
  or die("Cannot open $ARGV[0]/test_intron.fa\n");
open( ETEST, '<', "$ARGV[0]/test_exon.fa" )
  or die("Cannot open $ARGV[0]/test_exon.fa\n");

$mmOrder = $ARGV[2];    # Markov model order.

# INITIALIZE variables.
@handles          = ( 'ITRAIN', 'ETRAIN', 'ITEST', 'ETEST' );
$/                = '>';
$i                = 0;
$itrain_start     = $itrain_end = 0;
$etrain_start     = $etrain_end = 0;
$itest_start      = $itest_end = 0;
$etest_start      = $etest_end = 0;
$minimum_length   = 5 * ( $mmOrder + 1 );
$remaining_length = 0;

# READ sequence files.
foreach $handle (@handles) {

  if ( $handle eq 'ITRAIN' ) {
    $itrain_start = $i;
  } elsif ( $handle eq 'ETRAIN' ) {
    $etrain_start = $i;
  } elsif ( $handle eq 'ITEST' ) {
    $itest_start = $i;
  } else {
    $etest_start = $i;
  }

  while ( $record = <$handle> ) {
    @lines = split( "\n", $record );
    $header = shift(@lines);
    chomp(@lines);
    $sequence      = lc( join( '', @lines ) );
    $sequence       = trim($sequence);
    $current_length = length($sequence);

    # if too short, skip it.
    $remaining_length = $current_length - $frameshift;
    if ( $remaining_length < $minimum_length ) {
      next;
    } else {
      $nt_count = ( $sequence =~ tr/atgc// );
      if ( $current_length != $nt_count ) {
        next;
      } else {
        $sequences[$i] = substr($sequence, $frameshift, $remaining_length);
        $i++;
      }
    }
  }

  if ( $handle eq 'ITRAIN' ) {
    $itrain_end = $i;
  } elsif ( $handle eq 'ETRAIN' ) {
    $etrain_end = $i;
  } elsif ( $handle eq 'ITEST' ) {
    $itest_end = $i;
  } else {
    $etest_end = $i;
  }
  close($handle);
}

# INITIALIZE markov model variables.
%initE = %initI = %transE = %transI = ();
$initLen  = $mmOrder;
$transLen = $mmOrder + 1;
@scoresI  = @scoresE = ();

# Open output file.
open( OUT, '>', "$ARGV[1]" ) or die("Cannot open $ARGV[1].\n");

# TRAIN the classifier.
# Process introns.
for ( $i = $itrain_start ; $i < $itrain_end ; $i++ ) {
  $current_length = length( $sequences[$i] );
  for ( $pos = 0 ; $pos < ( $current_length - $initLen ) ; $pos++ ) {
    $key = substr( $sequences[$i], $pos, $initLen );
    $initI{$key}++;
    $sumInitI++;
  }

  for ( $pos = 0 ; $pos <= ( $current_length - $transLen ) ; $pos++ ) {
    $key = substr( $sequences[$i], $pos, $transLen );
    $transI{$key}++;
    $sumTransI++;
  }
}

# Process exons.
for ( $i = $etrain_start ; $i < $etrain_end ; $i++ ) {
  $current_length = length( $sequences[$i] );
  for ( $pos = 0 ; $pos <= ( $current_length - $initLen ) ; $pos++ ) {
    $key = substr( $sequences[$i], $pos, $initLen );
    $initE{$key}++;
    $sumInitE++;
  }

  for ( $pos = 0 ; $pos <= ( $current_length - $transLen ) ; $pos++ ) {
    $key = substr( $sequences[$i], $pos, $transLen );
    $transE{$key}++;
    $sumTransE++;
  }
}

# Convert occurrences to frequencies.
foreach $key ( keys(%initE) ) {
  $initE{$key} /= $sumInitE;
  $initI{$key} /= $sumInitI;
}

# TEST data.
# Process introns.
for ( $i = $itest_start ; $i < $itest_end ; $i++ ) {
  $current_length  = length( $sequences[$i] );
  $which           = $i - $itest_start;
  $scoresI[$which] = 0;

  $key   = substr( $sequences[$i], 0, $initLen );
  $probI = log( $initI{$key} );
  $probE = log( $initE{$key} );
  for ( $pos = 0 ; $pos <= ( $current_length - $transLen ) ; $pos++ ) {

    $key  = substr( $sequences[$i], $pos, $transLen );
    $keyA = substr( $key,           0,    $initLen ) . 'a';
    $keyG = substr( $key,           0,    $initLen ) . 'g';
    $keyC = substr( $key,           0,    $initLen ) . 'c';
    $keyT = substr( $key,           0,    $initLen ) . 't';

    # Calculate the denominators and log probabilities.
    $iDenom = $transI{$keyA} + $transI{$keyG};
    $iDenom += $transI{$keyC} + $transI{$keyT};
    $probI  += log( $transI{$key} / $iDenom );
    $eDenom = $transE{$keyA} + $transE{$keyG};
    $eDenom += $transE{$keyC} + $transE{$keyT};
    $probE  += log( $transE{$key} / $eDenom );
  }
  $scoresI[$which] = $probE - $probI;
}

# Process exons.
for ( $i = $etest_start ; $i < $etest_end ; $i++ ) {
  $current_length  = length( $sequences[$i] );
  $which           = $i - $etest_start;
  $scoresE[$which] = 0;

  $key   = substr( $sequences[$i], 0, $initLen );
  $probI = log( $initI{$key} );
  $probE = log( $initE{$key} );
  for ( $pos = 0 ; $pos <= ( $current_length - $transLen ) ; $pos++ ) {
    $key  = substr( $sequences[$i], $pos, $transLen );
    $keyA = substr( $key,           0,    $initLen ) . 'a';
    $keyG = substr( $key,           0,    $initLen ) . 'g';
    $keyC = substr( $key,           0,    $initLen ) . 'c';
    $keyT = substr( $key,           0,    $initLen ) . 't';

    # Calculate the denominators and log probabilities.
    $iDenom = $transI{$keyA} + $transI{$keyG};
    $iDenom += $transI{$keyC} + $transI{$keyT};
    $probI  += log( $transI{$key} / $iDenom );
    $eDenom = $transE{$keyA} + $transE{$keyG};
    $eDenom += $transE{$keyC} + $transE{$keyT};
    $probE  += log( $transE{$key} / $eDenom );
  }
  $scoresE[$which] = $probE - $probI;
}

# Output the classification scores.
print OUT "mm$mmOrder", "\n", join( ' ', @scoresI ), "\n",
  join( ' ', @scoresE ), "\n\n";
close(OUT);
```

References

[1] J. H. Do and D.-K. Choi, "Computational approaches to gene prediction.," *J Microbiol*, vol. 44, no. 2, pp. 137–144, 2006 Apr.

[2] R. Guigó, "Dna composition, codon usage and exon prediction." `http://www.pdg.cnb.uam.es/cursos/FVi2001/GenomAna/GeneIdentification/SearchContent/main.html`, February 7 2000.

[3] M. R. Brent, "How does eukaryotic gene prediction work?," *Nat Biotechnol*, vol. 25, no. 8, pp. 883–885, 2007 Aug.

[4] A. V. Lukashin and M. Borodovsky, "Genemark.hmm: new solutions for gene finding," *Nucleic Acids Res*, vol. 26, pp. 1107–15, Feb 1998.

[5] A. Lomsadze, V. Ter-Hovhannisyan, Y. O. Chernoff, and M. Borodovsky, "Gene identification in novel eukaryotic genomes by self-training algorithm," *Nucleic Acids Res*, vol. 33, no. 20, pp. 6494–506, 2005.

[6] V. Ter-Hovhannisyan, A. Lomsadze, Y. O. Chernoff, and M. Borodovsky, "Gene prediction in novel fungal genomes using an ab initio algorithm with unsupervised training," *Genome Res*, vol. 18, pp. 1979–90, Dec 2008.

[7] M. Borodovsky and J. Mcininch, "Genmark: Parallel gene recognition for both dna strands," *Computers & Chemistry*, vol. 17, no. 2, pp. 123–133, 1993.

[8] J. M. Bechtel, T. Wittenschlaeger, T. Dwyer, J. Song, S. Arunachalam, S. K. Ramakrishnan, S. S. Shepard, and A. Fedorov, "Genomic mid-range inhomogeneity correlates with an abundance of rna secondary structures," *BMC Genomics*, vol. 9, p. 284, 2008.

[9] A. Prakash, S. S. Shepard, J. He, B. Hart, M. Chen, S. P. Amarachintha, O. Mileyeva-Biebesheimer, J. Bechtel, and A. Fedorov, "Evolution of genomic

sequence inhomogeneity at mid-range scales," *BMC Genomics*, vol. 10, p. 513, 2009.

**equal contribution.*

[10] I. H. G. Consortium, "Finishing the euchromatic sequence of the human genome.," *Nature*, vol. 431, no. 7011, pp. 931–945, 2004 Oct 21.

[11] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, "Support vector machines and kernels for computational biology," *PLoS Comput Biol*, vol. 4, p. e1000173, 10 2008.

[12] G. Schweikert, A. Zien, G. Zeller, J. Behr, C. Dieterich, C. S. Ong, P. Philips, F. De Bona, L. Hartmann, A. Bohlen, N. Krüger, S. Sonnenburg, and G. Rätsch, "mgene: accurate svm-based gene finding with an application to nematode genomes," *Genome Res*, vol. 19, pp. 2133–43, Nov 2009.

[13] J. E. Allen, M. Pertea, and S. L. Salzberg, "Computational gene prediction using multiple sources of evidence.," *Genome Res*, vol. 14, no. 1, pp. 142–148, 2004 Jan.

[14] V. Shepelev and A. Fedorov, "Advances in the exon-intron database (eid)," *Brief Bioinform*, vol. 7, pp. 178–85, Jun 2006.

[15] A. Ruvinsky, S. T. Eskesen, F. N. Eskesen, and L. D. Hurst, "Can codon usage bias explain intron phase distributions and exon symmetry?," *J Mol Evol*, vol. 60, pp. 99–104, Jan 2005.

[16] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, June 1994.

[17] L. Hamel, "Model assessment with roc curves," in *The Encyclopedia of Data Warehousing and Mining*, Idea Group Publishers, 2009.

[18] N. A. Obuchowski, "Roc analysis," *American Journal of Roentgenology*, vol. 184, pp. 364–372, February 2005.

[19] A. Sboner, C. Eccher, E. Blanzieri, P. Bauer, M. Cristofolini, G. Zumiani, and S. Forti, "A multiple classifier system for early melanoma diagnosis," *Artif Intell Med*, vol. 27, pp. 29–44, Jan 2003.

[20] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on*, vol. 5, pp. 4104–4108 vol.5, 1997.

[21] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 2, pp. 171–195, 2008.

[22] S. Lee, H. Park, and M. Jeon, "Binary particle swarm optimization with bit change mutation," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E90-A, no. 10, pp. 2253–2256, 2007.

[23] S. Sonnenburg, B. S. Bernhard, P. Bennett, and E. Parrado-hernández, "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, p. 2006, 2006.

[24] C. W. Hsu, C. C. Chang, and C. J. Lin, *A practical guide to support vector classification.* Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2003.

[25] F. Provost, "Machine learning from imbalanced data sets 101," in *Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets*, 2000.

[26] J. C. Shepherd, "Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification," *Proc Natl Acad Sci U S A*, vol. 78, pp. 1596–600, Mar 1981.

[27] G. Bernardi, "Isochores and the evolutionary genomics of vertebrates.," *Gene*, vol. 241, no. 1, pp. 3–17, 2000 Jan 4.

[28] J. M. Bechtel, P. Rajesh, I. Ilikchyan, Y. Deng, P. K. Mishra, Q. Wang, X. Wu, K. A. Afonin, W. E. Grose, Y. Wang, S. Khuder, and A. Fedorov, "The alternative splicing mutation database: a hub for investigations of alternative splicing using mutational evidence.," *BMC Res Notes*, vol. 1, p. 3, 2008.

[29] J. M. Bechtel, P. Rajesh, I. Ilikchyan, Y. Deng, P. K. Mishra, Q. Wang, X. Wu, K. A. Afonin, W. E. Grose, Y. Wang, S. Khuder, and A. Fedorov, "Calculation of splicing potential from the alternative splicing mutation database.," *BMC Res Notes*, vol. 1, p. 4, 2008.

[30] T. Mitchell, "The discipline of machine learning.," Tech. Rep. CMU-ML-06-108, Carnegie-Mellon University, Pittsburg, Pennsylvania, 2006.

[31] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafe, A. Perez, and V. Robles, "Machine learning in bioinformatics," *Brief Bioinform*, vol. 7, pp. 86–112, March 2006.

[32] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann, second ed., 2006.

[33] Y. Nakamura, T. Gojobori, and T. Ikemura, "Codon usage tabulated from international dna sequence databases: status for the year 2000," *Nucleic Acids Res*, vol. 28, p. 292, Jan 2000.

[34] I. Grosse, P. Bernaola-Galván, P. Carpena, R. Román-Roldán, J. Oliver, and H. E. Stanley, "Analysis of symbolic sequences using the jensen-shannon divergence," *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 65, p. 041905, Apr 2002.