A Bioinformatics Tool for hnRNP-A1 Binding Site Prediction

By

Yanglong Mou

A Thesis

Submitted to the Faculty of Graduate Studies

In Partial Fulfillment of the Requirements

For the Degree of

Master in Science

Department of Biochemistry & Medical Genetics

University of Manitoba

Winnipeg, Manitoba

March, 2006

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A Bioinformatics Tool for hnRNP-A1 Binding Site Prediction

BY

Yanglong Mou

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

OF

Master of Science

Contents

Acknowledgements

This study was performed under the excellent professional guidance of Dr. Geoff Hicks (Department of Biochemistry & Human Genetics) and Dr. Peter Graham (Department of Computer Science), my supervisors. I would like to express my gratitude to them for their encouragement, guidance in scientific thinking and thoughtful discussion. I am thankful to my advisory committee members Dr. Ron Beavis (Department of Biochemistry and Medical Genetics) and Dr. Helen Cameron (Department of Computer Science) for their valuable comments and carefully reviewing of this thesis.

I would further like to extend my gratitude to the all my colleagues in the Manitoba Institute of Cell Biology, especially, Dr. Songyan Liu, for their help and suggestions during my study. I also thank the office staff of the Manitoba Institute of Cell Biology, Department of Biochemistry & Medical Genetics and Department of Computer Science for their availability and cooperation.

Finally, I would like to thank my wife Lisa for her inspiration and my son Eric (three and half years old) for the happiness he is bringing to my family.

Abstract

Heterogeneous nuclear ribonucleoprotein A1 (hnRNP-A1) has been shown to bind directly to pre-mRNA and regulate pre-mRNA splicing. hnRNP-A1 is the most comprehensively studied splicing factor of its class, and as such, provides the best opportunity to design and evaluate a bioinformatics tool for predicting splicing factor binding sites. We designed a program to predict the binding sites of hnRNP-A1 based on three mechanisms that are known to stabilize RNP-RNA interactions.

The process works in two steps. The first is to annotate known intron/exon boundaries for the query sequence using the *Ensembl* genome database as reference. The second step annotates RNA binding sites in pre-mRNA for the query sequence and then predicts potential hnRNP-A1 binding sites. Predicted hnRNP-A1 binding sites are identified based on the proximity of hnRNP-A1 binding site sequences to binding site sequences of either alternative splicing SR proteins, intronic/exonic splicing enhancers, branch point or multiple hnRNP-A1 sites that may allow the formation of an intron-exon-intron loop. Binding site sequence tables and proximity parameters are adjustable in the program code. We have also developed a web-interface for the program that will allow any scientist to predict potential hnRNP-A1 binding sites in a gene of interest which may help identify genes regulated by hnRNP-A1 or to elucidate the mechanisms of pre-mRNA splicing. The software program is written in Java.

To assess and optimize the program, we performed a predicted binding site analysis using a reference test set of genes that have previously been confirmed experimentally as bona fide hnRNP-A1 targets. The reference set includes genes coding

for *Mus musculus Src*, *Hras1*, Heterogeneous nuclear ribonucleoprotein A1 (*Hnrpa1*), Fibroblast Growth Factor Receptor-2 (*Fgfr2*), Adenylyl Cyclase Stimulatory G-protein Gαs (*Gnas1*), and the *Homo sapiens* genes *SRC*, *HRAS1*, Heterogeneous nuclear ribonucleoprotein (*HNRPA1*), Fibroblast Growth Factor Receptor-2 (*FGFR2*), Adenylyl Cyclase Stimulatory G-protein Gαs (*GNAS1*) and the tat gene of HIV (HIV-1 *tat*). Our results show that all hnRNP-A1 binding sites experimentally confirmed in these genes are correctly predicted using our software analysis. In addition, other possible novel hnRNP-A1 binding sites were identified and can be subsequently analyzed for secondary structure and experimental confirmation.

The software can be used genome wide to predict new A1 binding sites and to predict A1 regulated genes. Furthermore, the bioinformatics model established can be adapted to predict the splicing binding sites and target genes of other RNP proteins, including that of the proto-oncogene TLS.

The program can be accessed world wide at *http://140.193.242.11/*.

List of Tables

List of Figures

1. Introduction

1.1 The nucleotide sequence of a gene determines the amino acid sequence of a protein

One of the landmarks in biological research in the last century is the discovery that deoxyribonucleic acid (DNA) confers heritable properties of living organisms. In eukaryotic cells, the long, linear DNA molecule that contains millions of nucleotides is compacted together with DNA-binding proteins to form chromosomes. The amino acid sequence of a given protein is determined by DNA sequence (Alberts et al., 1994). As such, DNA molecules preserve the biological information that encodes the primary protein sequence. To translate the code from DNA to proteins, specific regions of DNA (genes) are first copied into a ribonucleic acid (RNA). This transcription results in a direct and complementary copy of the gene known as pre-mRNA (Figure 1). The pre-mRNA then undergoes a process called splicing during which non-coding portions of the pre-mRNA are removed. Coding portions of the pre-mRNA are then joined together to form messenger RNA (mRNA; Figure 2). In DNA, regions retained in mRNA after splicing are called exons, while those removed from mRNA are known as introns. mRNA serves as a template during protein synthesis. The message contained in the nucleotide sequence in mRNA is translated into the amino acid sequence of the protein, which determines the properties of the protein. Therefore, it is the nucleotide sequence in DNA that dictates the amino acid sequence and the properties of proteins (Alberts et al., 1994).

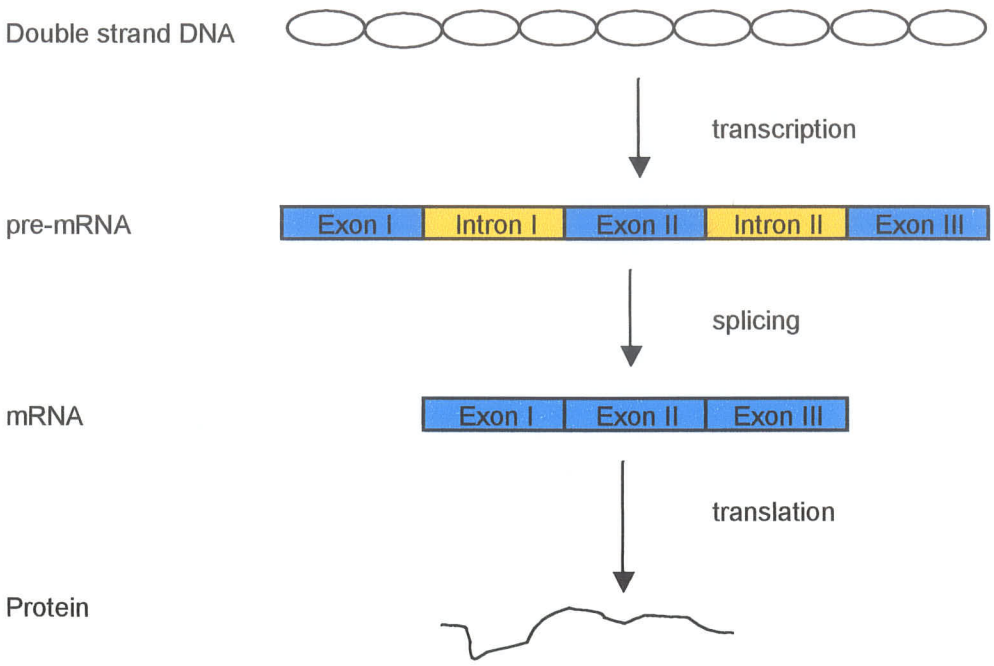

Figure 1: The nucleotide sequence of a gene determines the amino acid sequence of a protein. The nucleotide sequence is first transcribed into pre-mRNA which will undergo splicing to form mRNA. mRNA can then be used as a template for protein synthesis.

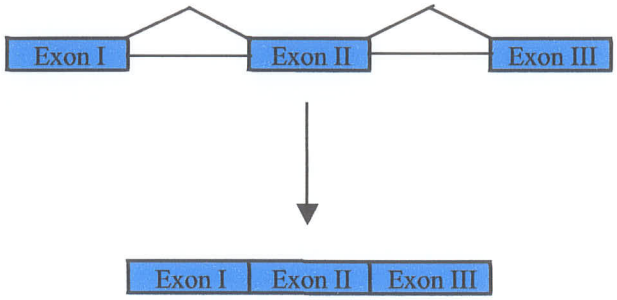

Figure 2: Pre-mRNA splicing. During the splicing, introns are removed from pre-mRNA and exons are retained.

It was estimated that our human chromosomes contained at least 30,000 genes after the publication of the human genome sequence (Venter et al., 2001). However, the number of different proteins observed is much greater than the number of genes contained in chromosomes. One major reason is that over 60% of genes can code for two or more proteins, often with different biological functions. The basis for this diversity is known as alternative splicing of the pre-mRNA to form different mRNAs.

1.2 Splicing signals and the splicing process in pre-mRNA

The splicing reaction is one of the methods that exist uniquely in eukaryotics for genetic regulation. During pre-mRNA splicing, intronic sequences are removed, while exonic sequences are retained. The selection of splice sites in the pre-mRNA is not random, but precisely controlled. This is very important. Mis-selection of splice sites can result in the addition or deletion of coding sequences, or pre-mature stops in the coding sequence, all of which will result in proteins with malfunctions or no functions.

1.2.1 Splicing signals and splicing-related factors

There are classic splicing signals (nucleotide sequences) in pre-mRNA in both 5′ and 3′ splice sites (Figure 3) (Smith and Valcarcel, 2000; Venter et al., 2001). Most of the nucleotides in these splicing signals are not conserved. Therefore, it is very difficult to predict 5′ and 3′ splice sites in pre-mRNA. However, some nucleotides are always present within the splicing signals. For example, at the 5′ splice site of the intron, the nearly invariant GU dinucleotide can be found, while AG dinucleotides always exist at the 3′ splice site of the intron. Upstream to this 3′ AG dinucleotide is the polypyrimidine

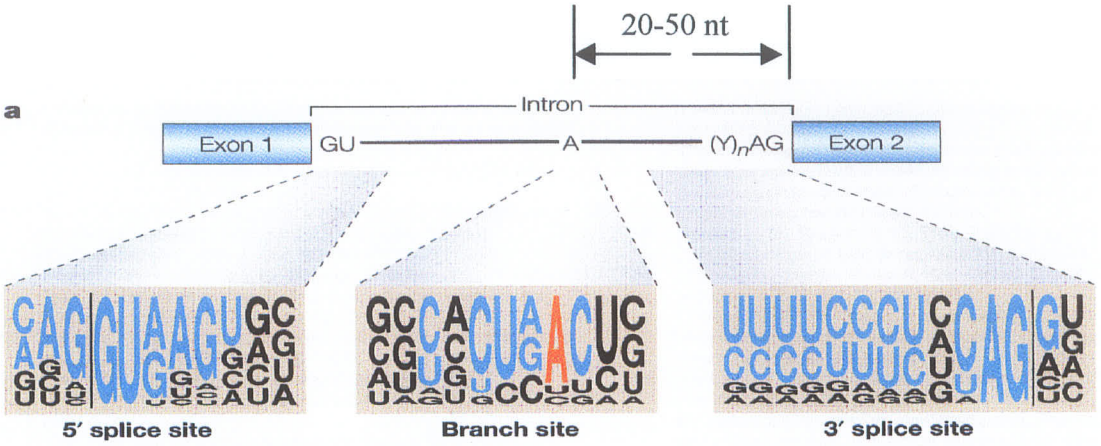

Figure 3: Classic splicing signals in both 5′ and 3′ splicing sites (modified from Cartegni et al, 2002). Figure shows that most nucleotides in a splicing signal are not conserved. However, in the 5' splicing site, GU nucleotides are conserved in position 2 and 3, while a residue A in the branchpoint which is located about 20-50 nucleotides upstream of the exon,, the polypyrimidine tract and dinucleotides AG can always be found in 3' splicing site.

tract (Yn). About 20 to 50 nucleotides upstream of the polypyrimidine tract is an A residue that serves as the splice branchpoint (Reed, 1996). Mutations in splicing sequences may cause changes in splicing patterns, sometimes resulting in disease. The splicing sequences are the binding sites for splicing-related factors (SRFs). The 5′ splice site interacts with the 5′ end of the U1 small nuclear RNA (U1 snRNA) (Burge and Karlin, 1997). The branchpoint sequence and its adjacent polypyrimidine tract are the binding sites for splicing factor 1 (SF1) and branchpoint binding protein (BBP) (Abovich and Rosbash, 1997; Arning et al., 1996; Berglund et al., 1997). The 65-kDa subunit of the dimeric U2 auxiliary factor (U2AF) has also been reported to interact with the polypyrimidine tract and, in some cases, the 35 kDa subunit of U2AF binds to the AG dinucleotide at the intron/exon junction (Guth et al., 1999; Wu et al., 1999). The intron borders are defined between the U1 snRNP complex at the 5′ splice site and the complex SF1-U2AF at the 3′ end of the intron. The physical interaction between these splicing-related factors and the nucleotides in the splicing signals binding sites is not strong. However, these binding proteins also interact with each other to form a network of interactions that further stabilize the binding of the proteins. For example, the SF1-U2AF complex at the 3′ splice site can interact with U1-associated proteins. At the same time, the SF1-U2AF complex binds to U1 snRNP/SRFs complexes downstream. Based on the interactions between splicing-related factors with splicing signals, internal exons are defined by the interactions between the SF1-U2AF complex at the 3′ splice site upstream of the exon and a U1 snRNP/SRFs complex at the downstream 5′ splice site (Robberson et al., 1990). Exon definition may also be mediated in part by SR proteins, which serve as a bridge between the U1 snRNP and the U2AF heterodimer (Wu and Maniatis, 1993).

1.2.2 The splicing process

Splicing takes place in a large macromolecular complex called a spliceosome that is composed of five small nuclear ribonucleoprotein particles (snRNPs) and 50-100 other proteins, many of which have not been identified yet. These proteins are not directly associated with snRNPs (Burge and Karlin, 1997).

The early steps of spliceosome assembly involve recognition of the consensus splicing signals at both ends of the intron as shown in Figure 3. First, the U1 snRNP, which contains both nucleotides and amino acids, binds to the 5′ splice site (characterized by the 5′ splicing signal) through base pairing between the 5′ splice signal and the nucleotide part of the U1 snRNA (Figure 4) (Black, 2000; Reed, 1996). Therefore, binding of U1 snRNA is sequence specific. At the same time, other splicing-related factors such as SF1 and U2AF bind to the 3′ splice site. At this stage, the spliceosome containing U1 snRNP and U2AF at the two intron ends is called the E (early) complex. This is followed by U2 snRNP binding to the branchpoint via base pairing. The complex at this stage is known as the A complex. The A complex is further joined by the U4, U5 and U6 snRNPs to form the B complex. After this, a complex rearrangement of the snRNPs in the B complex occurs to form the C complex, in which the U6 snRNP replaces the U1 snRNP to interact with the 5′ splice site and the U1 and U4 snRNPs are removed from the complex (Matlin et al., 2005).

Immediately following the formation of the C complex, two transesterification steps of the splicing reaction occur in the spliceosome (Figure 5). First, the 2′- hydroxyl group of the A residue at the branchpoint interacts with the phosphate of the G residue at the 5′ splice site; then, the 5′ exon is removed from the intron. This results in a detached

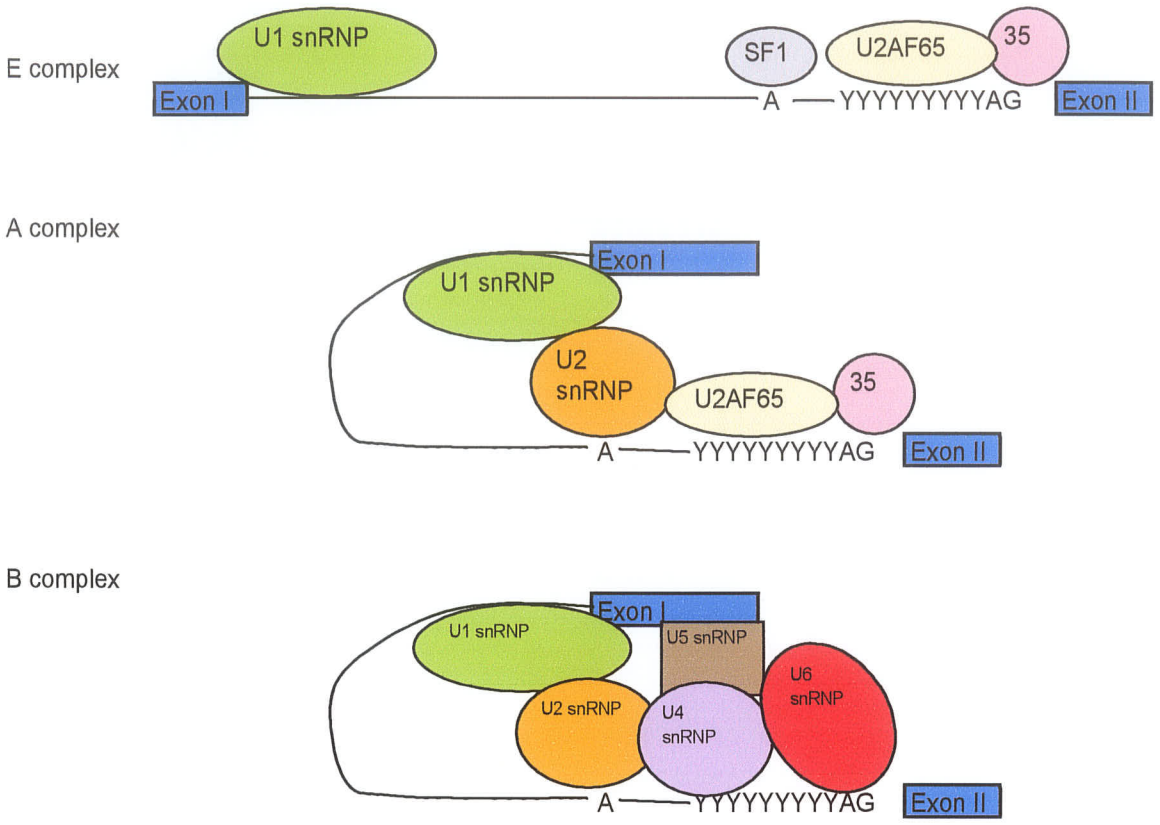

Figure 4: Processes of pre-mRNA splicing. Splicing occurs in a large macromolecular complex – the spliceosome. First, the E (early) complex is formed in which U1 snRNP binds to the 5′ splicing site and SF1, U2AF65, U2AF35 bind to the 3′ splicing site. Then, U2 snRNP binds to the branchpoint to cause a conformational change to form the A complex. The A complex is further joined by U4 snRNP, U5 snRNP and U6 snRNP, leading to a further conformational change of the complex, to form the B complex.

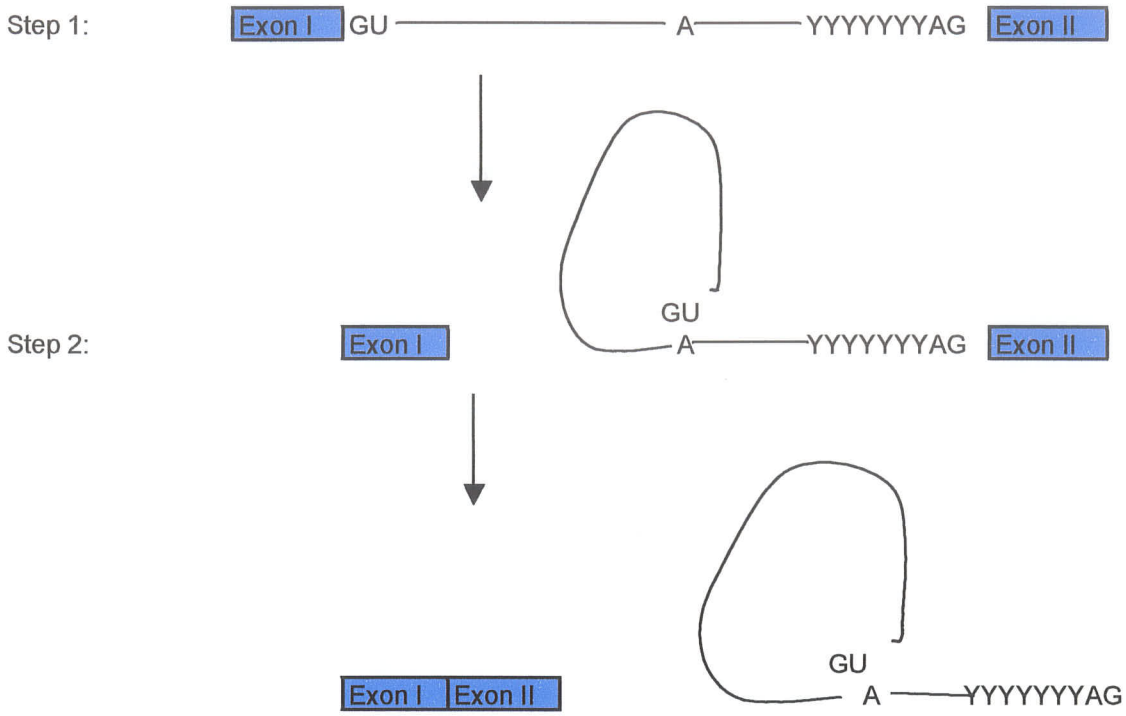

Figure 5: Two steps of trans-esterification during pre-mRNA splicing. In step 1, the 5' exon is cleaved from the intron, and the 2'-hydroxyl group of the A residue at the branch point joins with the phosphate of the G residue at the 5' splice site. In step 2, the phosphate at the 3' end of the intron is attacked by the 3'-hydroxyl of the detached exon, and the two exons are ligated together.

5′ exon and a lariat structure fragment containing an intron/3′-exon fragment. Next, the phosphate at the 3′ end of the intron is attacked by the 3′- hydroxyl of the detached exon, and the two exons are ligated to each other. At this stage, introns have been removed from pre-mRNA and only the spliced exons remain. The pre-mRNA then undergoes a series of additional modifications and becomes mRNA, which is ready to be used as a template for protein synthesis.

1.3 Alternative splicing in pre-mRNA

The splice sites selected during pre-mRNA splicing are not entirely constitutive. Alterations in splice site choice, or alternative splicing, may occur during pre-mRNA splicing. If this happens, it can have many different effects on the mRNA and thus their protein products. It is estimated that at least 60% of the human gene undergo alternative splicing (Modrek and Lee, 2002), and some gene transcripts have thousands of different splicing patterns (Black, 2000; Graveley, 2001). Alternative splicing determines which portions of the sequence in pre-mRNA should be included as coding sequence in mRNA. Therefore, alternative splicing gives rise to protein isoforms with different peptide sequences that may have different biological functions.

The exons that are always included in the final mRNA are defined as constitutive exons, while those that are not always included in the final mRNA are called cassette exons.

Several different patterns of alternative splicing have been found in eukaryotic pre-mRNA (Figure 6) (Cartegni et al., 2002). A cassette exon may be included or excluded in the final mRNA (exon skipping/inclusion). At times, alternative splicing

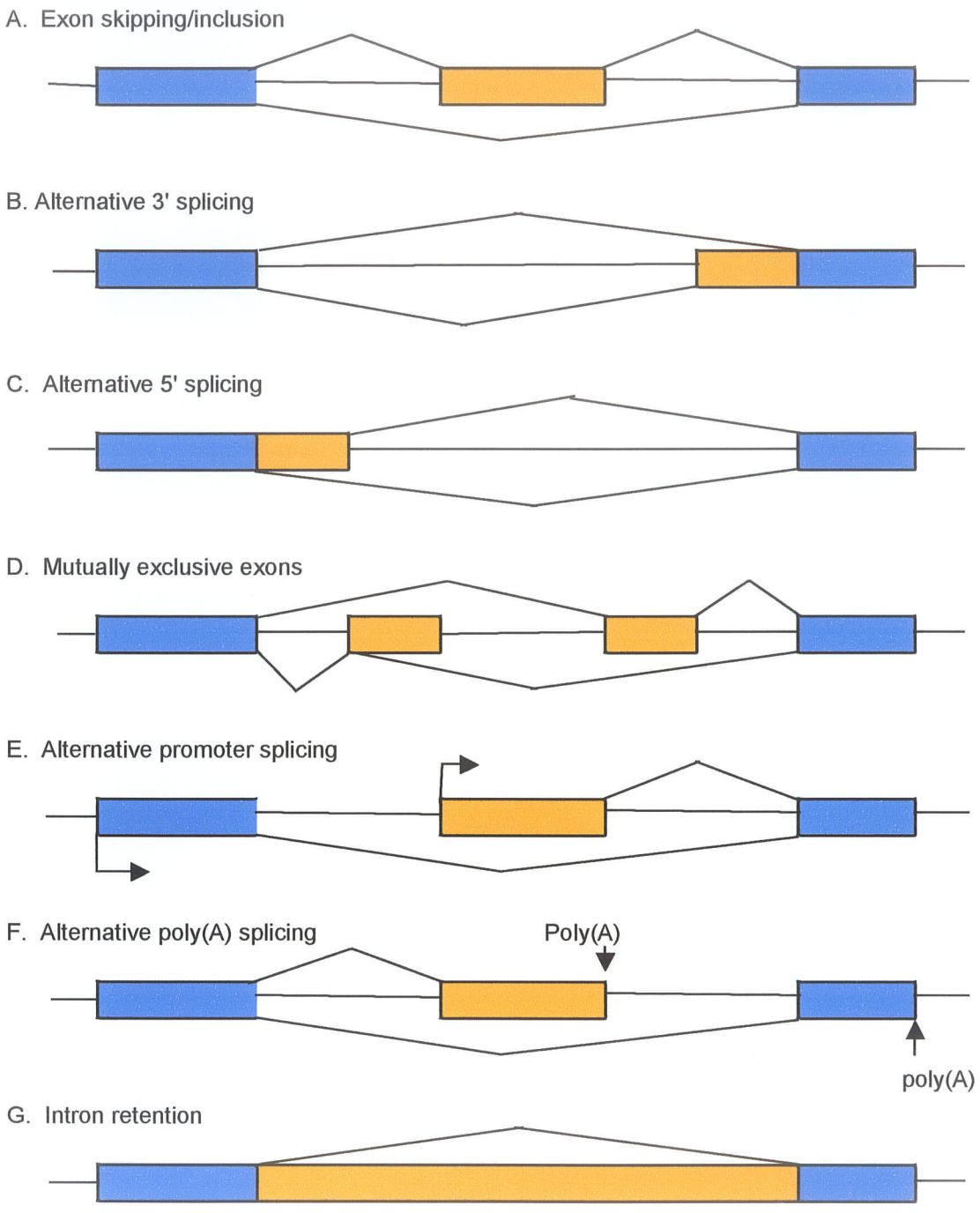

Figure 6: Patterns of alternative splicing. (A) A cassette exon (orange) may be excluded or included in the mRNA. (B, C) Alternative splicing occurs in part of the exon, causing alternative 5′ or 3′ splicing. (D) Two adjacent cassette exons are mutually excluded, resulting in only one of them included in the mRNA. (E, F) Alternative splicing happens in the 5′ promoter region or 3′ polyA region leading to alternative promoter splicing or alternative polyA splicing. (G) An intron is retained in the final mRNA.

happens within the exon itself. This can occur on either the 5′ or 3′ of the exon sequence, resulting in alternative 5′ or 3′ splicing. In some cases, two adjacent cassette exons are spliced such that only one of these cassette exons is included in the final mRNA (mutual exclusion). In other cases, alternative splicing occurs in the 5′ or 3′- most exon of the transcript, resulting in alternative promoters or alternative poly(A) sites (alternative promoter splicing or alternative poly(A) splicing). Alternative splicing may also happen when an intron is retained in or excised from the final mRNA (intron retention).

It should be noted that, in eukaryotic cells, multiple different patterns of alternative splicing mentioned above can be used in the pre-mRNA from the same gene.

1.4 Regulation of the splicing process

The regulation of the splicing process is very important. Regulation determines whether alternative splicing is, or which types of alternative splicing are, used for a given pre-mRNA. The regulation of the splicing process in a tissue or cell is very complex and many proteins have been reported to be involved in this regulation.

Pre-mRNA contains regulatory signals (regulatory elements) that are the binding sites for regulatory proteins. Based on their location in the exon or intron of a pre-mRNA sequence, the regulatory elements can be divided into exonic regulatory elements and intronic regulatory elements.

1.4.1 Exonic regulatory elements

The regulatory elements located in exons can be further divided into positive and negative exonic regulatory elements based on their effects on pre-mRNA splicing.

1.4.1.1 Positive exonic regulatory elements

There are many exonic splicing enhancers (ESEs) embedded within the coding region of the pre-mRNA. The binding to ESEs by proteins may enhance the binding of other splicing-related proteins to their binding sites, and thereby enhance (or having a positive effect on) splicing activities.

Different pre-mRNAs have different ESEs in their sequence. The identification of the ESEs is usually through sequence mutation in the exon or by SELEX (Systematic Evolution of Ligands by EXponential enrichment) (Tuerk et al, 1990). So far, many ESEs and their binding proteins have been identified. The mechanisms by which proteins binding to ESEs enhance splicing activities are still not clear. However, ESEs may stimulate U2AF or U2 binding to weak 3′ splice sites. They may also enhance the binding of U1 snRNP to 5′ splice sites (Bourgeois et al., 1999; Cote et al., 1999; Graveley, 2001; Selvakumar and Helfman, 1999). The best-studied family of ESE binding proteins is the SR protein family.

1.4.1.1.1 SR proteins

SR proteins are characterized by the presence of a common domain structure of one or two RNA recognition motifs (RRM) in the N terminal and an RS domain containing repeated arginine/serine dipeptides (hence, *SR* proteins). Both RRM and RS domains are important during RNA splicing. RRM domains mediate sequence-specific binding to RNA while the RS domain seems to serve as a bridge to connect with other protein complexes such as U1 snRNP/splicing-related factor complex (Shen et al., 2004a;

Shen et al., 2004b). Proteins belonging to the SR protein family include SRp20, SRp30, SRp40, SRp55, SRp 70, SC35, 9G8 and SF2/ASF (Bourgeois et al., 1999; Cote et al., 1999; Graveley, 2001; Selvakumar and Helfman, 1999). ESEs have been shown to be the binding sites for specific SR proteins. SELEX *in vitro* has been used to characterize high-affinity binding sequences of these SR proteins. Many functional ESEs have been identified by the combination of SELEX *in vitro* with a S100 complementation assay (Table 1) (Cartegni et al., 2002).

1.4.1.2 Negative exonic regulatory elements

Exons may also contain splicing silencer or repressor elements called exonic splicing silencers (ESSs). The binding to ESSs by negative regulators may cause inhibition of pre-mRNA splicing activities. ESSs are less well characterized than exonic splicing enhancers (ESEs). Many proteins have been shown to interact with ESSs. The mechanism by which binding to ESSs by negative regulators inhibits pre-mRNA splicing is still not fully understood. However, in some situations, ESSs overlap with ESEs that are recognized by SR proteins. This suggests that direct competition by inhibitory factors could lead to splicing silencing (Zhu et al., 2001).

1.4.2 Intronic regulatory elements

Similar to regulatory elements in exons, introns contain many sequences that serve as regulators for pre-mRNA splicing. They are the binding sites for the splicing-related factors. These binding sites can be located at a branchpoint, polypyrimidine tract

Table 1: SR proteins and their corresponding binding site sequence[1]

Protein	High-affinity binding site[2]	Functional ESE
SRp20[3]	WCWWC	GCUCCUCUUCC
	CUCKUCY	CCUCGUCC
SC35[4]	AGSAGAGUA	GRYYMCYR
	GUUCGAGUA	UGCYGYY
	UGUUCSAGWU	
	GWUWCCUGCUA	
9G8[5]	(GAC)n	
	ACGAGAGAY	
SF2/ASF[6]	RGAAGAAC	CRSMSGW
	AGGACRRAGC	
SRp40[7]	UGGGAGCRGUYRGCUCGY	YRCRKM
SRp55[8]		YYWCWSG

[1] Reviewed in (Cartegni L et al, 2002).
[2] M=A/C, R=A/G, W=A/U, Y=C/U, S=C/G, K=G/U
[3] (Schaal T et al, 1999; Calvio C et al, 1995; Lou H et al, 1998).
[4] (Tacke R et al, 1995; Schaal T et al, 1999; Liu H et al, 2000; Cavalo Y et al, 1999).
[5] (Cavaloc Y et al, 1999; Schaal T et al, 1999).
[6] (Liu H et al, 1998; Tacke R et al, 1995).
[7] (Liu H et al, 1998; Tacke R et al, 1995).
[8] (Liu H et al, 1998).

or 5′ splice site. The binding sites can also be located hundreds of nucleotides downstream of the 5′ splice site or upstream of the 3′ splice site.

1.4.2.1 Positive intronic regulatory elements

Several sequences that are called intronic splicing enhancers (ISEs) have been found (Black, 1992; Forch et al., 2000; Modafferi and Black, 1997); however, the proteins that bind to these ISEs have been less well characterized than ESEs. One ISE element is the sequence containing UGCAUG, especially when this hexanucleotide appears in duplication. Binding to this region by regulatory proteins has been shown to enhance the splicing of exons in many transcripts such as c-src, fibronectin and calcitonin transcripts. Deletion or mutations in this sequence will lead to an inhibitory effect in the splicing. In mouse neuronal LA-N-5 cells, this hexanucleotide is located downstream of the related exon N1, and it exerts its effect in combination with other multiple small elements (Black, 1992; Guo and Kawamoto, 2000; Hedjran et al., 1997; Huh and Hynes, 1993; Huh and Hynes, 1994; Modafferi and Black, 1997). As yet, the protein that binds to UGCAUG has not been identified and the mechanisms of splice regulation through ISEs have not been elucidated.

1.4.2.2 Negative intronic regulatory elements

Like exonic splicing silencers (ESSs), there are also negative intronic regulatory sequences called intronic splicing silencers (ISSs). For example, some SR proteins such as SRp30c have been shown to bind to intron splice sites in pre-mRNA of the heterogenous nuclear ribonucleoprotein A1 (hnRNP-A1) gene and inhibit splicing

(Simard and Chabot, 2002). Binding of polypyrimidine-tract-binding protein (PTB) to the 3′ splice site has been reported to inhibit splicing activities in many transcripts such as rat β-tropomyosin (Mulligan et al., 1992) and smooth muscle-specific exon in α-actinin (Southby et al., 1999). Perhaps the best-studied protein that binds to ISSs is hnRNP-A1. hnRNP-A1 has been used as a model to elucidate how splicing-related factors influence splicing activities (Blanchette and Chabot, 1999; Tange et al., 2001; Zhu et al., 2001). The characteristics of hnRNP-A1 and the mechanisms of hnRNP-A1- mediated inhibitory effects during pre-mRNA splicing are discussed below.

1.5 Properties of hnRNP A1 and the mechanisms of its inhibitory effects on pre-mRNA splicing

1.5.1 Structure of hnRNP A1

hnRNP-A1 is a member of the heterogenous nuclear ribonucleoprotein (hnRNP) family. The hnRNP-A1 protein is located in the nucleoplasm of interphase cells, but is also found to shuttle between the nucleus and the cytoplasm. hnRNP-A1 contains a consensus RNA-recognition motif (RRM) and a glycine-rich auxiliary domain at the carboxyl terminus (Biamonti et al., 1989; Buvoli et al., 1990). The RRM domain is found in many RNA-binding proteins located in the nucleus, cytoplasm, and cytoplasmic organelles. RRM contains two consensus sequences called RRM1 and RRM2 and each sequence contains about 90 amino acids. RRM1 and RRM2 are separated by about 30 amino acids (Figure 7). X-ray crystallography shows that each RRM motif has a β1-α1-β2-β3-α2-β4 structure (Figure 8) (Gorlach et al., 1992; Wittekind et al., 1992). The four β strands form an anti-parallel β sheet that packs against the two α helices. Nuclear

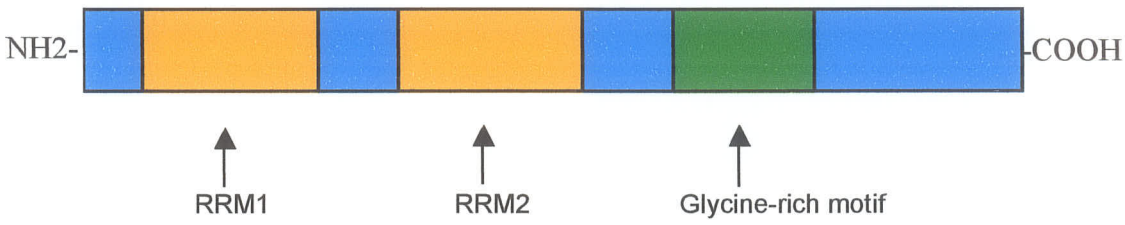

Figure 7: Domain structure of hnRNP-A1. hnRNP-A1 contains two RNA recognition motifs (RRMs) in the N-terminal and one glycine-rich motif in the C-terminal. RRM is involved in RNA binding, while the glycine-rich motif stabilizes interactions with RNA and other proteins.

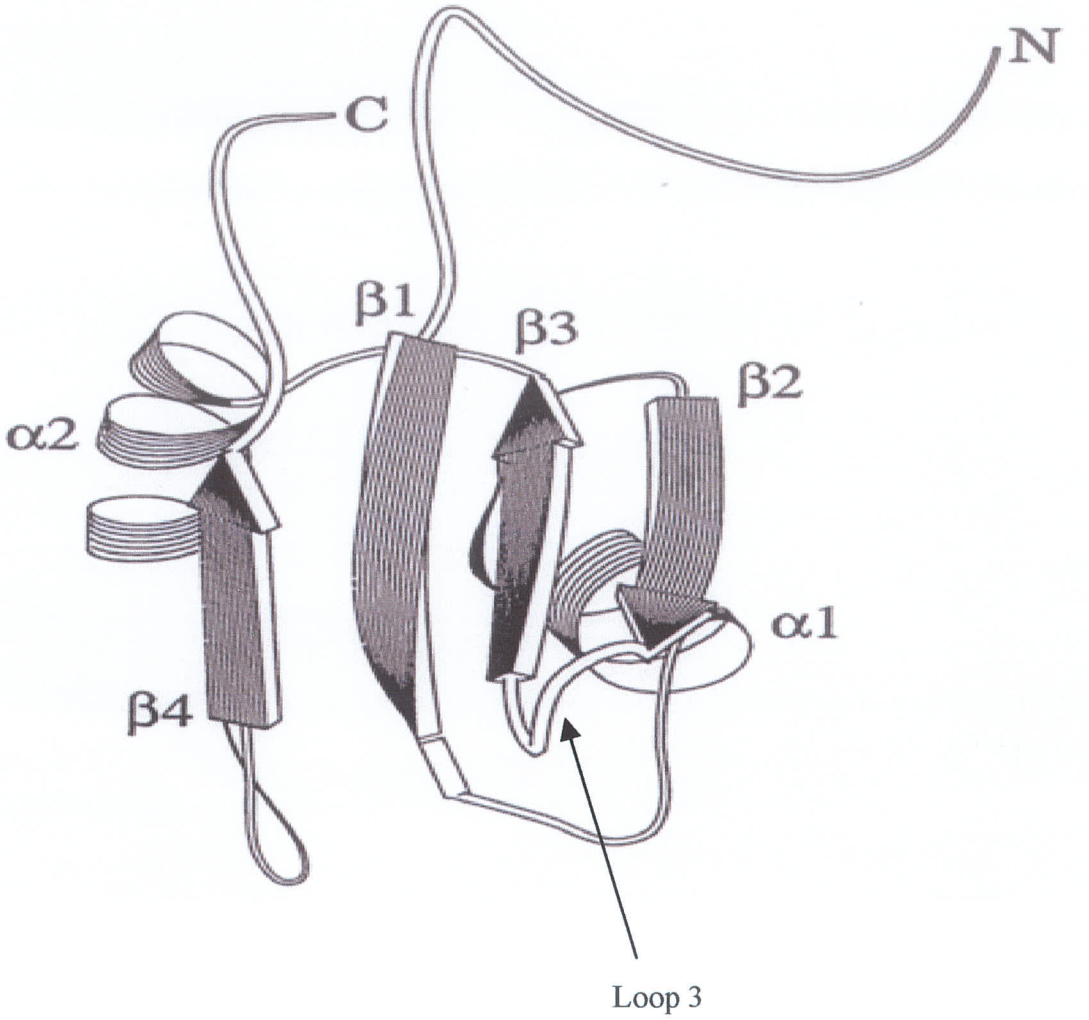

Figure 8: Schematic ribbon drawing of the hnRNP C RRM (residues 2-94) (modified from Gorlach M et al, 1992). The RRM motif has a β1-α1-β2-β3-α2-β4 structure. The four β strands form an anti-parallel β sheet that packs against the two α helices. The RNA binding site in RRM is located in loop 3 between β2 and β3.

Magnetic Resonance (NMR) data from RRM show that the hydrophobic inner surface of the domain from this β1-α1-β2-β3-α2-β4 structure forms the core structure of the RRM, while the exposed surface serves as a platform to which the RNA binds. The most variable region is located in the loop between β2 and β3, which determines RNA binding specificity. Chemical shifts of the residues in the β sheet region will result in significant perturbation of RNA binding (Gorlach et al., 1992; Hoffman et al., 1991).

1.5.2 Mechanisms by which hnRNP-A1 mediates inhibitory effects during pre-mRNA splicing

hnRNP-A1 has been shown to bind to the pre-mRNA of many transcripts including HIV tat (Amendt et al., 1995; Zhu et al., 2001), Fibroblast Growth Factor Receptor 2 (FGFR-2) pre-mRNA (Del Gatto and Breathnach, 1995), c-H-ras pre-mRNA (Guil et al., 2003), and mediate inhibitory effects during pre-mRNA splicing. hnRNP-A1 has also been shown to be able to regulate the splicing of pre-mRNA coding for hnRNP-A1 itself (Chabot et al., 1997).

At least three mechanisms have been proposed for hnRNP-A1-mediated splicing repression (Figure 9). (1) hnRNP-A1 specifically binds to ESEs. This specific binding by hnRNP-A1 to ESEs leads to nucleating the assembly of additional hnRNP-A1 molecules along the pre-mRNA, which results in the formation of a zone of hnRNP-A1, thus repressing the binding of SR proteins (Zhu et al., 2001). (2) hnRNP-A1 specifically binds to the branchpoint to inhibit the binding of the SF1 protein. This will lead to blocking of U2 snRNP binding to the branchpoint (Tange et al., 2001) and the assembly of the spliceosome. (3) hnRNP-A1 can also bind to intronic binding sites on both sides

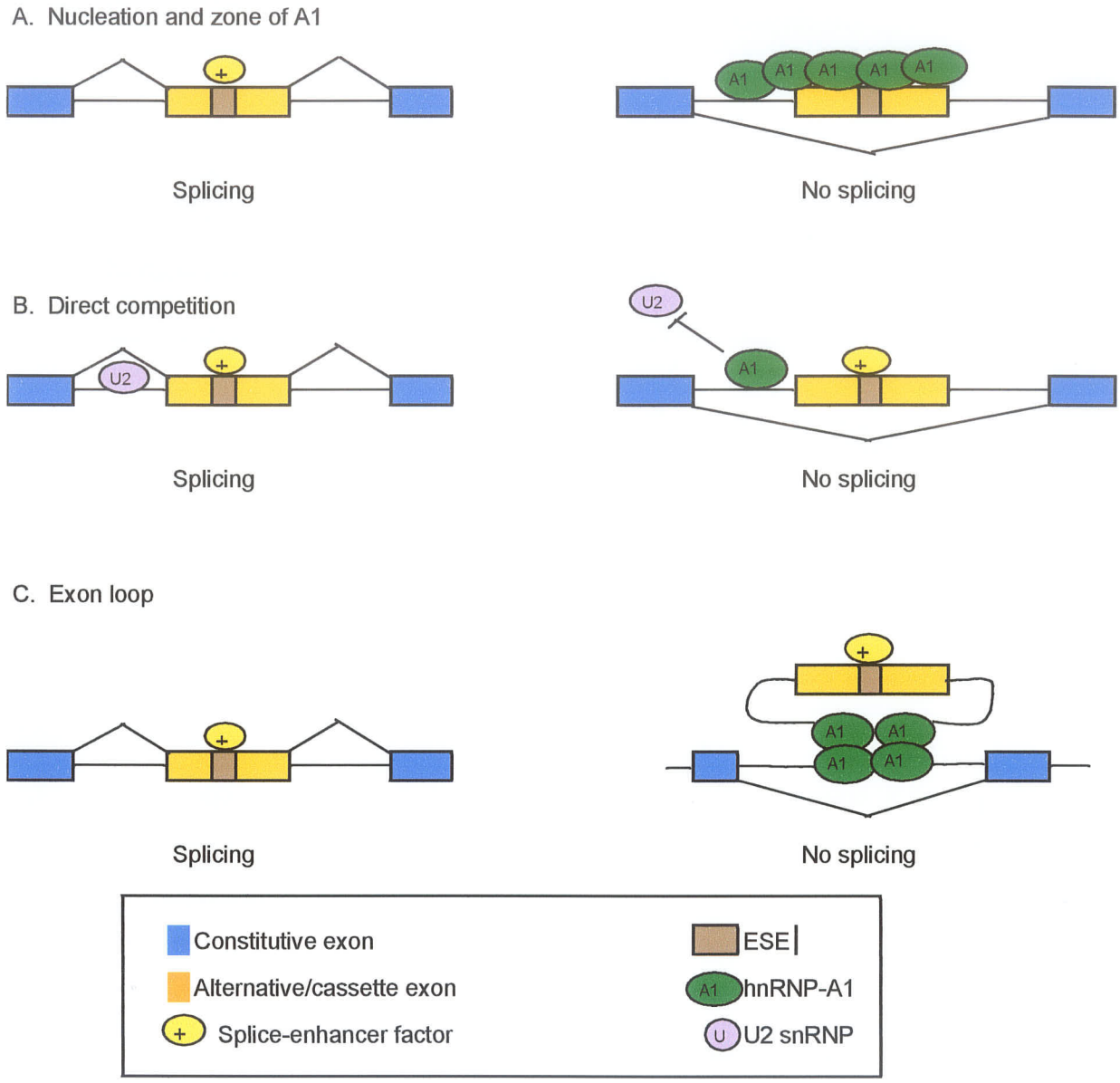

Figure 9: Mechanisms by which hnRNP-A1 regulates pre-mRNA splicing. The left and right sides of the figure show splicing in the absence and presence of hnRNP-A1 respectively. (A) hnRNP-A1 binding to the exon leads to the assembly of additional hnRNP-A1 molecules along the pre-mRNA, which represses the binding of splice-enhancing factors such as SR proteins. (B) hnRNP-A1 binds to the branchpoint and blocks U2 snRNP binding to pre-mRNA, thereby preventing exon definition. (C) Multiple hnRNP-A1s bind to the intron and multimerization results in an intron-exon-intron loop.

of the exon. The binding of hnRNP-A1 to these sites can cause multimerization of hnRNP-A1 on both sides of the exon, resulting in the formation of a loop which loops the exon out and causes exon skipping (Blanchette and Chabot, 1999). In addition, hnRNP-A1 has also been reported to bind directly to intronic splicing silencers (ISSs) and exonic splicing silencers (ESSs). Several of the binding sequences in ISSs (GGCAGUGAGGGAGGCGAGGG) (Guil et al., 2003) and ESSs (UAGUGAA, CUAGACUAGA) (Caputi et al., 1999; Marchand et al., 2002) have been identified. Thus, hnRNP-A1 seems to use multiple mechanisms to regulate pre-mRNA splicing.

The nucleotide sequence recognized by hnRNP-A1 is not conserved. However, by using a systematic evolution of ligands by exponential enrichment (SELEX), UAGGGU/A has been identified to be the high affinity sequence (Burd and Dreyfuss, 1994). Other sequences can also be recognized by hnRNP-A1 (Del Gatto and Breathnach, 1995; Guil et al., 2003; Pollard et al., 2002). Table 2 shows the experimentally validated hnRNP-A1 binding sequences and these binding sequences are expressed in a pictogram (Figure 10). As with most RNA binding proteins, the binding sequence for hnRNP-A1 is degenerate and may vary in different sequences. This increases the difficulty in predicting hnRNP-A1 binding sites in pre-mRNA.

1.6 Summary of splicing regulation

Splice sites are not randomly selected by cells during pre-mRNA splicing, but are guided by specific sequences (splicing signals) in pre-mRNA. The splicing signals in both the 5′ and 3′ splice sites are not completely conserved. They serve as the binding sites for splicing-related factors such as SF1 and U2AF. Pre-mRNA may also undergo

Table 2: Experimentally validated hnRNP-A1 binding site sequences

1[1]	U	A	G	G	G	U
2[2]	U	A	G	G	G	A
3[3]	U	A	G	G	A	U
4[4]	U	A	G	G	A	A
5[5]	U	A	G	A	G	U
6[6]	U	A	G	A	G	A
7[7]	U	A	G	A	A	U
8[8]	U	A	G	A	A	A
9[9]	U	A	G	G	C	A
10[10]	U	A	G	G	C	U
11[11]	U	A	G	A	U	U
12[12]	U	A	G	U	G	A
13[13]	C	A	G	G	G	A
14[14]	U	A	G	G	U	A
15[15]	U	A	G	G	G	C
16[16]	U	A	G	G	A	G
17[17]	G	A	G	G	G	A
18[18]	U	A	G	A	C	U
19[19]	U	A	G	A	A	G

[1] (Burd C et al, 1994).
[2] (Burd C et al, 1994).
[3] (Hutchison S et al, 2002).
[4] (Hutchison S et al, 2002).
[5] (Hutchison S et al, 2002).
[6] (Hutchison S et al, 2002).
[7] (Hutchison S et al, 2002).
[8] (Hutchison S et al, 2002).
[9] (Tange T et al, 2003).
[10] (Chabot B et al, 1997; Burd C et al, 1994).
[11] (Blanchette M et al, 1999).
[12] (Tange T et al, 2001).
[13] (Burd C et al, 1994; Tange T et al, 2001).
[14] (Burd C et al, 1994).
[15] (Del Gatto F et al, 1995).
[16] (Rooke N et al, 2003).
[17] (Guil S et al, 2003).
[18] (Zhu J et al, 2001).
[19] (Caputi M et al, 1999; Marchand V et al, 2002).

Figure 10: hnRNP-A1 binding site sequences shown in pictogram. The pictogram shows the relative frequency of nucleotide residues in an hnRNP-A1 RNA binding site sequence, based on Table 1. For pre-mRNA, T residues are transcribed as U residues.

alternative splicing during the splicing reaction. Alternative splicing is a very important feature in eukaryotic cells. It enables cells to use one gene to encode more than one protein with different biological functions. The pre-mRNA splicing activities can be influenced by special sequences (elements) in both exons and introns of the pre-mRNA which are the binding sites for splicing regulatory factors. How these splicing regulatory factors influence splicing activities has not been fully elucidated yet. However, many factors, especially SR proteins, have been shown to be able to enhance splicing activities through binding to exonic splicing enhancers (ESEs) which are specific sequences in an exon. SR proteins not only bind to RNA, but also function as bridges to link one protein complex to another protein complex. So far hnRNP-A1 is the best studied protein that exhibits repression affecting splicing activities. hnRNP-A1 is able to bind intronic splicing silencers (ISSs) which may be located hundreds of nucleotides away from the splice site. Three mechanisms by which hnRNP-A1 mediates repression effects in splicing activities have been proposed. The binding sequence of hnRNP-A1 is not completely conserved although hnRNP-A1 shows high binding affinity to UAGGGU/A in *in vitro* SELEX. This makes it difficult to predict hnRNP-A1 binding sites in RNA.

1.7 Analysis of alternative splicing by bioinformatics

In recent years, bioinformatics has penetrated into almost every aspect of biological research. It has also been used in analyzing alternative splicing to predict alternative splice forms and to identify alternative splicing elements.

1.7.1 Prediction of alternative splice forms

Expressed sequence tags (EST) are derived from fully processed mRNA. EST sequence databases have provided a broad source for predicting alternative splice forms (Modrek and Lee, 2002). By looking for the differences among ESTs derived from the same gene, large insertions or deletions in ESTs can easily be identified (Kan et al., 2001; Modrek et al., 2001). Alternative splice forms can also be found by alignment of DNA and transcript sequence data. Clark and Thanaraj constructed a data set of transcript-confirmed exons and introns from 2793 genes, 798 of which were found to have multiple isoforms (Clark and Thanaraj, 2002). In recent years, microarray data has been used to explore splicing variations. mRNA isoforms from as little as 10-100 pg of total cellular RNA can be detected. The identified candidate splice variants can be used to compare to the alternatively spliced transcripts to identify alternative splice forms (Hu et al., 2001; Yeakley et al., 2002). By using statistical analysis of the microarray data, Johnson *et al.* have discovered about 800 exon-skipping events that were not previously detected using ESTs (Johnson et al., 2003).

1.7.2 Identifying alternative splicing regulatory elements

Bioinformatics has been shown to be an effective tool for identifying alternative splicing regulatory elements such as exonic and intronic splicing enhancers (ESEs, ISEs) or exonic and intronic splicing silencers (ESSs, ISSs). By statistical analysis of intron-exon and splice site composition, Fairbrother *et al.* investigated hexanucleotide sequences that are enriched in exons and are near weak splice sites. Ten classes of predicted ESEs have been found. Among these ten classes, five of them match with identified ESEs, and

the rest are novel ESEs. All ten ESEs are further confirmed by their enhancing effect in pre-mRNA splicing by experiments with point mutations in their sequences (Fairbrother et al., 2002). Brudno *et al.* used a computational approach to find possible ISEs involved in cell-type-specific alternative splicing. They compared a kilobase of intronic sequence on either side of 25 brain-specific internal alternatively spliced exons with those of constitutive exons. The hexanucleotide UGCAUG was found significantly enriched in the downstream intron of cassette exons. This suggests that this hexanucleotide may play a role in regulation of exon splicing (Brudno et al., 2001). These results are consistent with the previous finding that hexanucleotide UGCAUG serves as an ISE during c-src pre-mRNA splicing (Modafferi and Black, 1997).

Taken together, bioinformatics approaches can facilitate exploring splicing and alternative splicing on a genome-wide scale. Investigators can now compare and contrast vast genomic, mRNA, and EST databases to find new alternative splicing forms, to predict splicing regulatory elements and to develop profiles of how alternative splicing is regulated. With the aid of established biological databases and software, much raw data, including the regulation of pre-mRNA splicing and alternative splicing, can be obtained even before biological experiments are performed.

1.8 The TLS Proto-oncogene and Thesis Rationale

The alternative splicing feature in eukaryotic cells allows a new level of complexity in which splicing may play a major role in the regulation of gene expression or that one eukaryotic gene may encode multiple proteins with different functions.

However, the mechanisms by which alternative splicings are themselves regulated in eukaryotic cells are still not very clear. The major focus of our laboratory work is the proto-oncogene *TLS*, a ribonuclear protein found translocated in several human cancers (Delva et al., 2004; Meissner et al., 2003; Rapp et al., 2002). TLS was also independently discovered as the gene encoding the high nuclear ribonucleoprotein P2 protein (hnRNP-P2), which was identified as a component of a large protein splicing complex assembled on adenovirus pre-mRNA (Calvio et al., 1995). Several additional lines of evidence support a functional role for TLS in splicing. TLS is engaged in a complex with the hnRNPs A1 and C1/C2 (Zinszner et al., 1994), and TLS is also associated with several spliceosomal small nuclear ribonucleoproteins (Hackl et al., 1994). Furthermore, TLS has recently been reported to be able to regulate pre-mRNA splicing (Hallier et al., 1998; Lerga et al., 2001). More detailed reports have shown that TLS favors the selection of the distal 5′-splice site during E1A pre-mRNA splicing in vivo (Hallier et al., 1998) and the TLS RNA binding sequence has been identified using SELEX to be GGUG (Lerga et al., 2001). While most of this data is correlative and based on in vitro studies, these data do indicate that the RNA binding activity of TLS may play an important role in the regulation of RNA splicing. Even more interesting is the identification of specific *T*LS-*A*ssociated *SR* proteins (TASR's; (Clinton et al., 2002)). This would suggest a more direct role for TLS in the regulation of alternative splicing for specific subsets of target genes, as opposed to a more perfunctory role in the general process of pre-RNA splicing.

The identification of TLS-specific target genes that may be directly regulated by alternative splicing could provide critical insights into the mechanisms by which the

proto-oncogene initiates or drives cancer. Unfortunately, the majority of the tools developed for genome-wide screening for potential target genes are not well suited for identifying alternatively-spliced gene products. With the complete sequence of the genome available, bioinformatic approaches have been developed to aid in the prediction of target genes for oncogenic DNA binding transcription factors (Li et al., 2005). These approaches are facilitated by the fact that transcription factors bind with high affinity and high sequence specificity within a very short and discrete promoter region of a gene. RNA binding proteins, on the other hand, bind with low affinity and low sequence specificity, and range over the entire length of the gene – making a simple binding site search approach unlikely to provide meaningful results. However, the transcription factor approach works because the identified binding sites can be placed in a meaningful context – a relatively short promoter sequence. We rationalized that if we could define a correspondingly similar context for RNA binding sites, we should be able to develop a bioinformatics tool that could predict potential alternative binding sites.

For the purpose of testing our hypothesis, we chose to model our splicing program on hnRNP-A1. hnRNP-A1 is in the same RNA binding super family as TLS but has the distinct advantage that it is one of the best studied RNA binding proteins and the mechanisms used by hnRNP-A1 to regulate splicing have been partially elucidated. More importantly, hnRNP-A1 RNA binding site sequences and hnRNP-A1 regulated target genes have been identified and experimentally validated. Therefore, hnRNP-A1 would be an ideal model to establish proof of principle for developing a bioinformatic approach.

In this thesis, we developed software to predict hnRNP-A1 binding sites in pre-mRNA based on a reference set of known hnRNP-A1 binding site sequences and target genes. We have also developed a web-interface for the program that will allow any scientist to predict potential hnRNP-A1 binding sites in a gene of interest and which may help identify genes regulated by hnRNP-A1 or to elucidate the mechanisms of pre-mRNA splicing. Furthermore, we hope that the model established in this thesis can eventually also be used to predict the binding sites and genes in which alternative splicing is regulated by TLS.

2. Materials and Methods

2.1 Programming tool and software used:

The programs for the hnRNP-A1 binding site prediction tool were written using the Java programming language and run under Sun Java™ 2DK, Sun Forte™ for Java™, Community Edition, Version 1.0. The Apache HTTP server version 1.3 was downloaded from http://www.nusphere.com/download.pp.ide.htm and was used as the server for the web portal to the software. The web interface was written in HTML (Hyper Text Markup Language). PHP (PHP: Hypertext Preprocessor) programming was used to invoke the Java programs. The BLAST (Basic Local Alignment Search Tool) program was downloaded from the National Center for Biotechnology Institute (NCBI) website at http://www.ncbi.nlm.nih.gov/BLAST/ and incorporated into the system. All parameters for BLAST were set to defaults. If a BLAST query sequence for a given gene of interest is longer than 1,000 nt, only the first 1,000 nt of the sequence is used for BLAST.

2.2 Annotated intron/exon position file and genomic sequence:

Annotated intron/exon position files of human and mouse were downloaded from the *Ensembl* website at http://www.ensembl.org/Multi/martview. Briefly, ***Ensembl 36 and Homo sapiens genes (NCBI35)*** or ***Mus musculus genes (NCBIM34)*** were selected, and ***next*** was clicked, then ***next*** was clicked again. ***Structure*** was selected in ***Select the Attribute Page***. ***Ensembl Gene ID*** in ***Ensembl Attributes of GENE***, ***Exon Start (Chr bp)*** and ***Exon End (Chr bp)*** in ***Exon Attributes*** **of** ***EXON***, and ***Text, comma separated*** in ***Select the output format*** and ***None*** in ***File compression*** were selected and then ***export***

was clicked. The annotated intron/exon file was downloaded and saved to the local hard drive.

Annotated human and mouse genomic sequence were downloaded from http://www.ensembl.org/Multi/martview. Briefly, ***Ensembl 36*** and ***Homo Sapiens genes (NCBI35)(NCBI35)*** or ***Mus musculus genes (NCBIM34)(NCBIM34)*** were selected and then ***next*** was clicked twice. ***Sequences*** in ***Select the Attribute Page***, and ***Unspliced (Transcript)*** in ***Type of Sequence to Export (all in 5'-3' direction)*** were selected and then ***export*** was clicked. The exported file was formatted to the request of BLAST.

2.3. Reference gene sequence set

Homo sapiens Adenylyl Cyclase Stimulatory G-protein $G\alpha_s$ (*GNAS1*) (ENSG00000087460), fibroblast growth factor receptor -2 (*FGFR2*) (ENSG00000066468), *SRC* (ENSG00000197122), *Mus musculus Gnas1* (ENSMUSG00000027523), *Src* (ENSMUSG00000027646) and *Fgfr2* (ENSMUSG00000030849) were exported in FASTA format (Lipman et al., 1985) from http://www.ensembl.org/Homo_sapiens/textview. Briefly, ***Homo sapiens*** or ***Mus musculus*** databases were selected. Then the *Ensembl* ID was entered and ***Search*** was clicked. A list of information related to the entered *Ensembl* ID is displayed. When the *Ensembl* ID in ***Ensembl Gene*** was clicked, more detail information will be shown. Then ***Export data on*** the left side of the screen was clicked, ***FASTA format text file*** was selected and ***Continue*** button was clicked. Finally ***Text*** **in** ***Output format*** was selected and the ***Continue*** button was clicked. The genomic DNA sequence for the gene of interest is then shown in FASTA format. *Homo sapiens HNRPA1* gene sequence

(X12671), *HRAS1* gene sequence (J00277), *Mus musculus Hnrpa1* gene sequence (U65316), *Hras1* (Z50013) and HIV-1 *tat* gene sequence (NC_001802) were exported in FASTA format from http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed by searching ***Nucleotide*** in ***for gene ID***. When gene ID and source were selected, the FASTA format of the DNA sequence will be exported by selecting ***FASTA*** in ***Display***.

2.4 Design of the underlying algorithm and my bioinformatic tool

2.4.1 Processes for hnRNP-A1 binding site prediction

The processes used in this thesis for hnRNP-A1 binding site prediction are shown in Figure 11. Briefly, when a genomic DNA query sequence for a given gene of interest and species are submitted by the user, an execution file for the BLAST program is called to launch BLAST. The BLAST program will search for the query sequence against the complete annotated genome sequence, previously downloaded from the *Ensembl* website. After user selects the *Ensembl* ID of the query sequence based on the BLAST result, the *Ensembl* ID of the query sequence and its corresponding start and end positions are stored. These data are used to search the annotated intron/exon file downloaded from *Ensembl* to annotate the exon and intron positions in the query sequence. In the next step, the query sequence is searched to identify the location of validated hnRNP-A1 binding site sequences (Table 2), the location of SR proteins binding site sequences (Table 1), and the location of ESSs (UAGUGAA, CUAGACUAGA) and ISSs (GGCAGUGAGGGAGGCGAGGG). Finally, the program uses the criteria (explained in detail in the next section) to identify which hnRNP-A1 binding site location will be reported as a predicted hnRNP-A1 binding site.

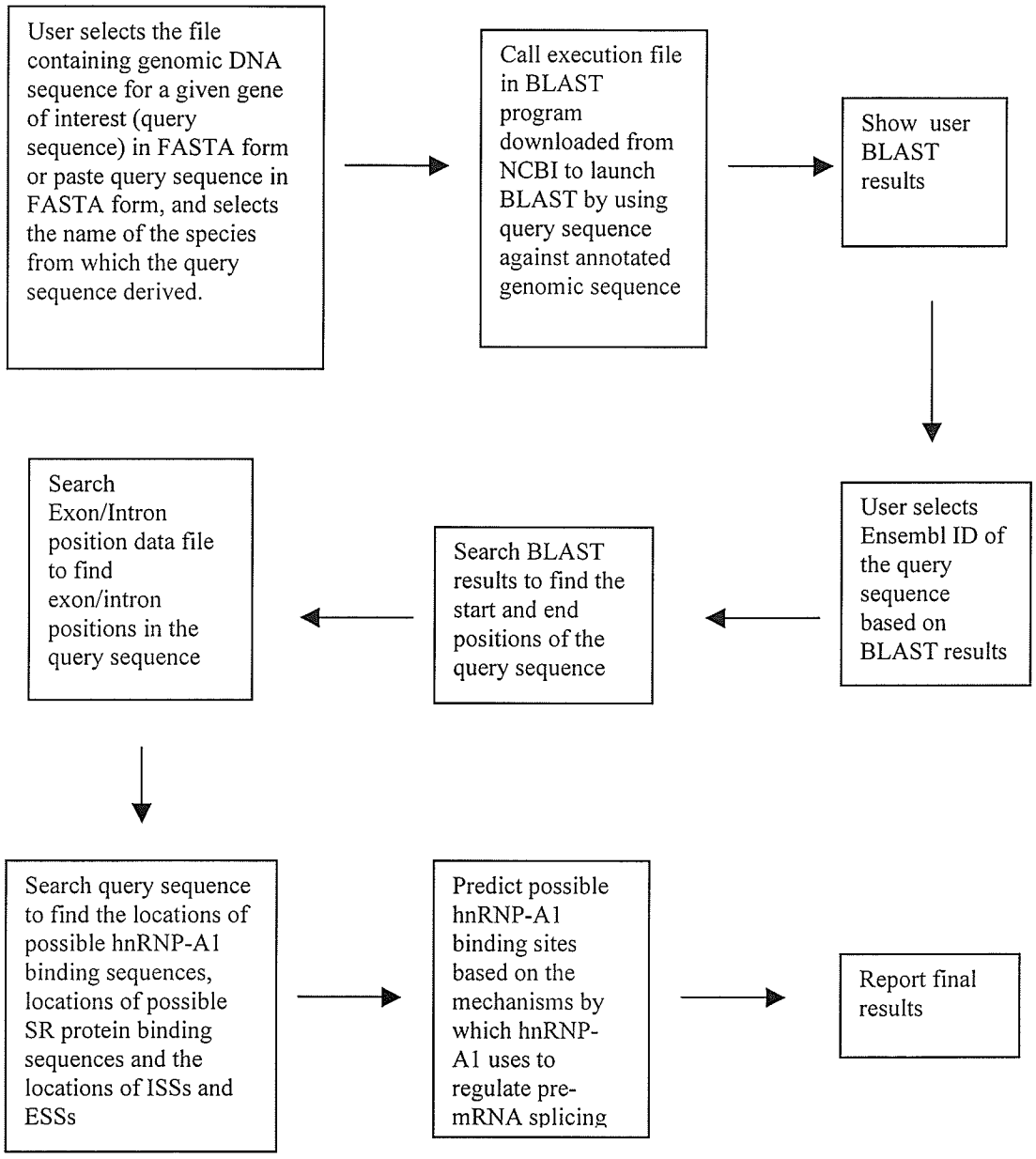

Figure 11: Process for hnRNP-A1 binding site(s) prediction used in this project.

2.4.2 Criteria used for hnRNP-A1 binding site prediction

Each hnRNP-A1 binding site sequence is assessed using the following criteria to determine whether it is a predicted hnRNP-A1 binding site.

(1) Does the location of the hnRNP-A1 binding site sequence overlap with the location of an SR protein by at least 1 nt?

(2) Is the location of the hnRNP-A1 binding site sequence overlap with the location of ISSs or ESSs by at least 1 nt?

(3) Is the location of the hnRNP-A1 binding site sequence within 20 – 50 nt upstream of an exon (Branchpoint)?

(4) Are there at least two hnRNP-A1 binding site sequences within 500 nt both upstream of the 5′- and downstream of the 3′-sides of an exon?

All hnRNP-A1 binding site sequences that meet any one of the four criteria will be reported as a predicted hnRNP-A1 binding site.

2.4.3 Algorithms used and programming strategy for hnRNP-A1 binding site prediction

2.4.3.1 Search Ensembl ID in BLAST report

When the user selects the *Ensembl* ID of the query sequence, the program searches BLAST report for the *Ensembl* ID which matches the one selected by user. The BLAST report (see Figure 13) lists the *Ensembl* IDs of the highest scoring matches and is located as individual lines beginning with the ">" mark. A brief description of the gene directly follows its *Ensembl* ID, including the gene start and end position, is separated by the symbol '|'. A human or a mouse gene *Ensembl* ID contains 17 or 19 characters

respectively, and the last 2 characters are related to the version number. Therefore, the algorithm used in this program searches the first 16 (">" plus the first 15 characters of the user selected *Ensembl* ID for a human gene) or 18 (">" plus the first 17 characters of the user selected *Ensembl* ID for a mouse gene) characters in each line of the BLAST report until the user-selected ID is found. Once the matched Ensembl ID is found, the gene start position and end position can be located by the class StringTokenizer in Java.

2.4.3.2 Query sequence intron/exon position annotation

Query sequence intron/exon position annotation is based on the annotated intron/exon position file downloaded from the *Ensembl* database. The file contains the Ensembl ID, the start position of the exon and the end position of the exon. Each item is separated by ','. To annotate the intron/exon position of the query sequence, the program searches the first 15 or 17 characters of each line for the user-selected *Ensembl* ID in the file. Once the matched *Ensembl* ID is found, the intron/exon positions are then annotated using the class StringTokenizer in Java.

2.4.3.3 Searching for hnRNP-A1 binding site sequences, SR binding site sequences, ISSs and ESSs in the query

The hnRNP-A1 binding site sequences (Table 2), SR binding site sequences (Table 1), ISSs (GGCAGUGAGGGAGGCGAGGG) and ESSs (UAGUGAA, CUAGACUAGA) each contains no more than 20 characters and the average length of a query sequence is 10-15,000 nt (the average length of a gene). As such, searching for hnRNP-A1, SR protein binding site, ISS and ESS sequences in the query will not

consume much CPU time. Therefore, a simple Brute-Force algorithm is used in this project.

2.4.3.4 Algorithm for data sorting

The program requires data in several arrays to be sorted. The hnRNP-A1 binding site sequence is the array that contains the most amount of the data. The average length of a query sequence is 10-15,000 nt, and is expected to contain an average of 46 -70 hnRNP-A1 binding site sequences. Therefore, we required only a simple data sorting algorithm, and chose the Bubble sorting algorithm.

2.4.3.5 Programming strategy

After the program annotates the intron/exon positions of the query sequence, the results are stored in an array called ***exonPosition***. The program then searches the query to find all possible hnRNP-A1 binding site sequences (Table 2), ISS (GGCAGUGAGGGAGGCGAGGG) and ESS (UAGUGAA, CUAGACUAGA) sequences, and determines their locations within the query sequence. The locations are stored in arrays named ***locationArray***, ***iss*** and ***ess,*** respectively. The locations of SR protein binding site sequences in the query are stored in arrays named for the corresponding SR protein. All of the arrays are sorted ascendant by Bubble sorting.

The program then uses the criteria mentioned in section *2.4.2 (Criteria used for hnRNP-A1 binding site prediction)* to determine the predicted hnRNP-A1 binding sites from the ***locationArray***, which contains the locations of all the hnRNP-A1 binding site sequences. The predicted hnRNP-A1 binding sites are stored in an array called

indication. The contents in the array ***indication*** are sorted by Bubble algorithm in ascending order and redundant content is removed. The final content of the ***indication*** array is reported as the hnRNP-A1 binding sites. Finally, the program prompts the user to analyze the provided sequences further for secondary structure confirmation at http://www.bioinfo.rpi.edu/applications/mfold/old/rna .

2.5 Computer used:

The computer used in executing the software developed in this thesis was a Pentium 4 running at 1.83 GHz with a 60 GB hard drive and 512 MB in RAM.

3. Results

3.1 The User Interface for the bioinformatics tool

We have implemented the hnRNP-A1 binding site prediction tool as a web-based program to allow greater access and ease of use for the scientific community at large (refer to Figure 12; or point your browser to *http://140.193.242.11/*). The process works in two steps. The first step annotates known intron/exon boundaries for the query sequence. The second step annotates RNA binding sites in pre-mRNA for the query sequence and then predicts potential hnRNP-A1 binding sites.

The definition of the intron/exon boundaries utilizes *Ensembl* genome annotation, and therefore, the critical aspect of the first step is for the user to identify the correct *Ensembl* gene for their query. Users can either paste their query sequence in FASTA format or click "Browse" to select a file that contains the query sequence in FASTA format. Radio buttons allow the user to identify whether the query sequence is human- or mouse-derived. Clicking the "Submit" button launches a local BLAST program to align the query sequence to the annotated genomic sequence file exported from *Ensembl*. Sequences from other species can be predicted by manually entering intron/exon positions (not described on the web page).

The run time for BLAST may vary from a few seconds to several minutes depending on the length of the query sequence. When completed, a second window will appear showing the BLAST results. The first ten best matches (high BLAST Score and low Expectation cutoff value) are shown with their sequence *Ensembl* ID in a Combo Box. The results include *Ensembl* ID, chromosome number, strand direction, BLAST score, E value and sequence alignment. Users can easily select the correct *Ensembl* ID of

University of Manitoba

Manitoba Institute of Cell Biology

Mammalian Functional Genomic Centre

Useful Bioinformatics Tools

Home

hnRNP-A1 BINDING SITE(S) PREDICTION TOOL

This software is still in its TEST STAGE. We welcome people to use it and feed back your comments and suggestion.

Two steps to predict hnRNP-A1 binding sites in your query sequence:

(1) Paste your query sequence in FASTA format or select a file containing your query sequence in FASTA format, and click Submit. A BLAST results will be popped out in a new window.

(2) Select ENSEMBL ID of your query sequence and click Continue. The results of hnRNP-A1 binding site prediction will be shown up.

Paste your sequence in FASTA format here

or Select your file:

Browse...

Select Speices:

Human Mouse

Submit Reset

Contact:
Dr. Geoff Hicks
Manitoba Institute of Cell Biology 675 McDermot Ave. Rm. ON5029 Winnipeg, MB R3E 0V9
Tel: (204) 787-2133, Lab: (204) 787-2167 Fax: (204) 787-2190 hicksgg@cc.umanitoba.ca.

This website is created and maintained by Dr. Yanglong Mou. You can also send email to mouy@cc.umanitoba.ca.

This website was created in July 2005.

Figure 12: The web page of this project. The web page is located at http://www.140.193.242.11. Users can paste their query sequence or select a file containing their query sequence in FASTA format, and then select the species from which their query sequence is derived. The query sequence is then submitted for BLAST analysis using the "Submit" button.

their query sequence, if already known, or refer to displayed information of chromosome number, BLAST score, E value, and sequence alignment to assist in identifying the *Ensembl* ID of the query sequence (Figure 13). At this time, users select the correct *Ensembl* ID for their query sequence and click the "Continue" button.

The results for the entire second step of the process appear in a new screen in less than 3 seconds. The report shows the query sequence and results for the predicted hnRNP-A1 binding sites. The location of each predicted binding site is identified and marked with the possible mechanism used by hnRNP-A1 to regulate splicing. Each predicted binding site is followed by a pre-mRNA sequence that includes the predicted hnRNP-A1 binding site flanked by 500 nt of the upstream and downstream intronic sequences. Intron and exon sequences are clearly indicated using either small or capital letters respectively. The provided sequence also allows the user to analyze the pre-mRNA sequence for secondary structure using the linked m-fold program ((Zuker, 2003); http://www.bioinfo.rpi.edu/applications/mfold/old/rna). hnRNP-A1 binding sites located in a stem-loop structure would indicate stronger support of a valid binding site (Varani, 1995). Figure 14A shows an example of the secondary structure derived from part of the third predicted hnRNP-A1 binding region (2867-3350). The stem-loop structure from 2878-3004 containing UAGAGA is highlighted in Figure 14B. Users can also view the hnRNP-A1 binding sequence distribution and predicted binding site(s) distribution in the query sequence.

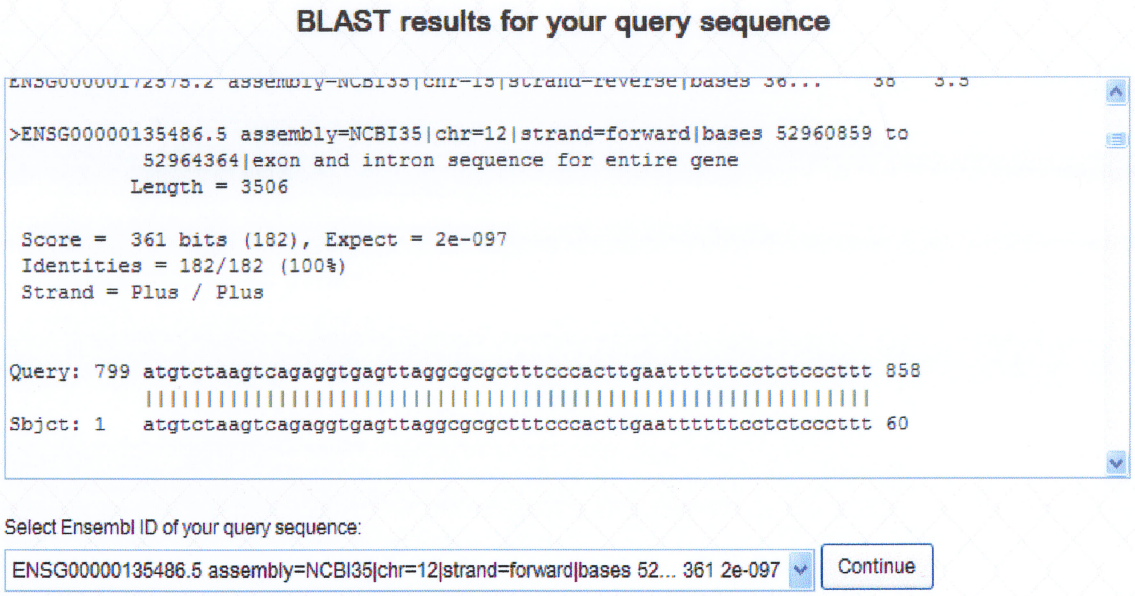

Figure 13: BLAST result window. The result shown contains *Ensembl* ID, chromosome number, strand direction, BLAST score, and E-value. Sequence alignments for the top ten matches are shown. Users can use this information to select the correct *Ensembl* ID for their query.

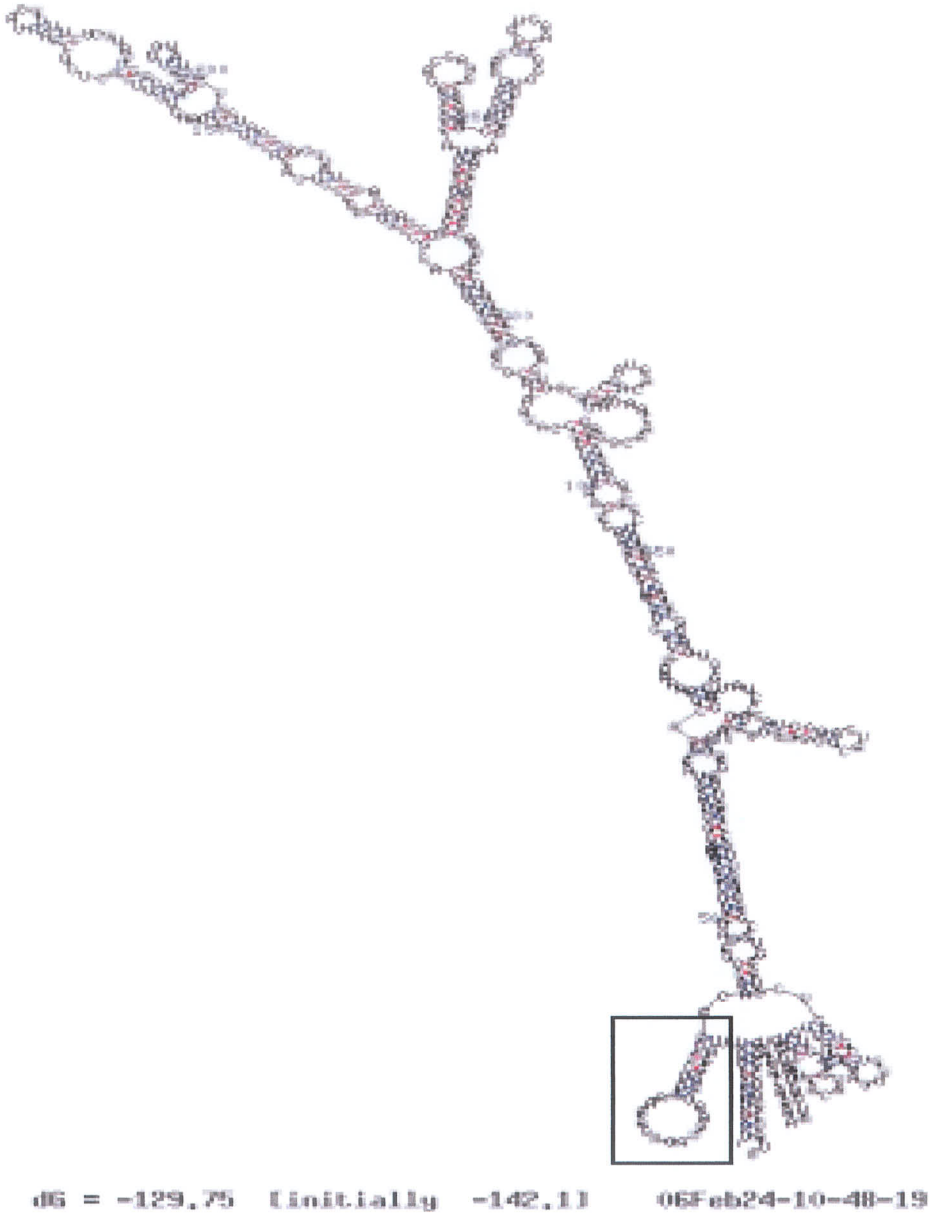

Figure 14A: Secondary structure of human hnRNP-A1 pre-mRNA (region 2879 nt – 3356 nt) by m-fold. A predicted hnRNP-A1 binding site forms stem loop structure, highlighted in box and enlarged in Figure 14B.

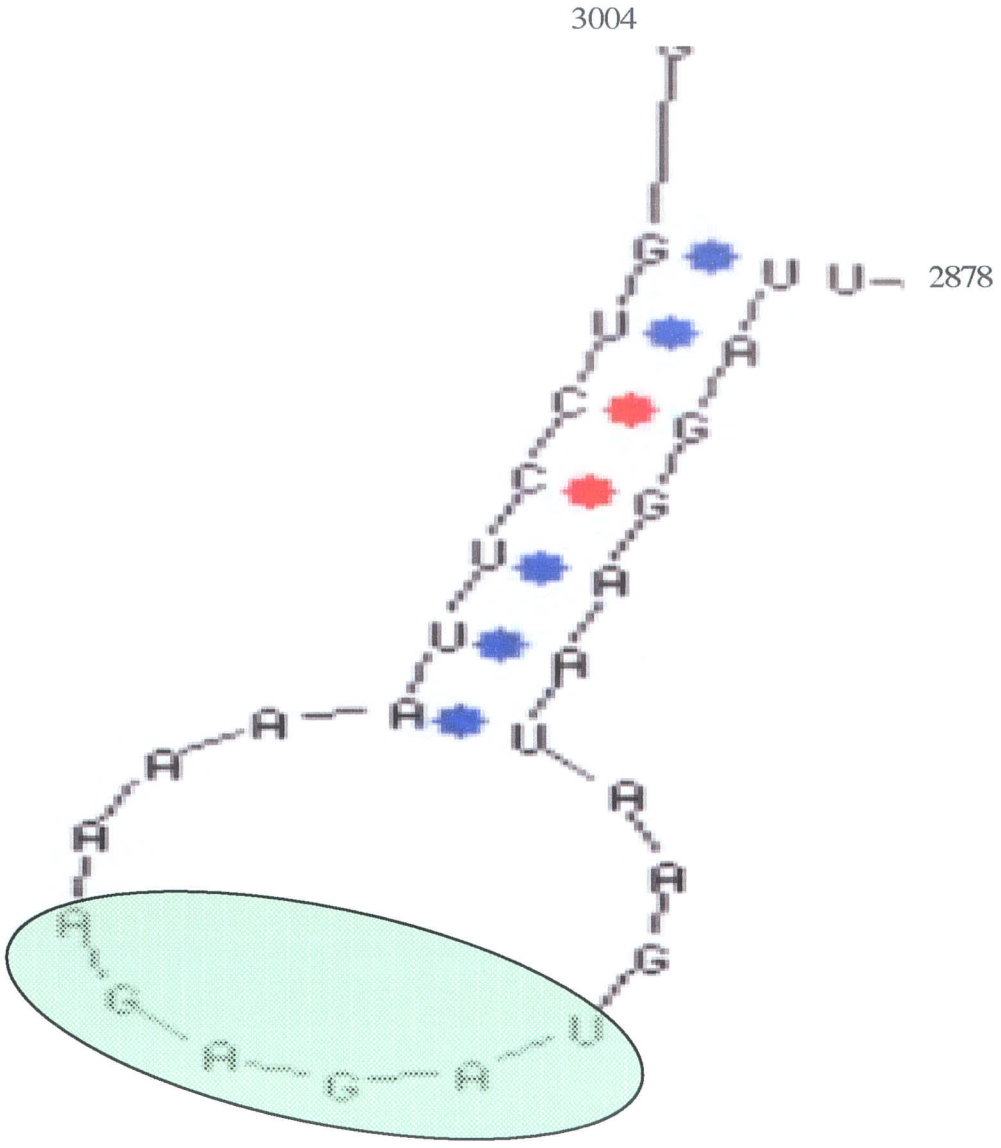

Figure 14B: hnRNP-A1 stem-loop structure. A predicted hnRNP-A1 binding site sequence, UAGAGA, from 2878-3004 nt, forms the loop structure (green oval).

3.2 Parameter settings

Biologically relevant binding of hnRNP-A1 to RNA molecules is sequence-dependent; however, it is well established that the formation of inter-stabilizing complexes with other RNA binding proteins is equally important for functional activity. The hnRNP-A1 binding site predictor program is based on identifying RNA-binding protein sites within a genomic sequence and then predicting candidate hnRNA-A1 binding sites based on the proximity of that site to RNA binding sites for functionally relevant RNA binding proteins. The parameters for my program, therefore, fall into two classes: first, the defining of specific RNA binding site sequences; and second, the defining of "proximity" between identified hnRNP-A1 binding sites and identified binding sites for functionally relevant proteins. For RNA binding sites, we have chosen to include only those that have been validated experimentally. These are detailed in Tables 1 and 2. The rationale for selecting functionally relevant RNA binding proteins has been detailed extensively in the introduction. To reduce the number of false positive predictions, parameters for "proximity" were chosen conservatively, and, again, based only on experimentally validated models of RNA splicing.

For the experimental results detailed in the next two sections, the program had the following default parameter settings:

i) hnRNP-A1 binding site proximity to an SR binding site is set at "not less than one nucleotide overlap."

ii) hnRNP-A1 binding site proximity to an ISS or ESS consensus binding site is set at "not less than one base pair overlap."

iii) hnRNP-A1 binding site proximity to the spliceosome initiator complex is set at the branchpoint, "20-50 nucleotides upstream of an annotated exon."

The complete program code can be found in Appendix 1. It is important to note that these parameters can be readily changed to allow for the analysis of additional experimental data sets.

3.2.1 Optimizing parameter settings for the hnRNP-A1 intron-exon-intron loop model.

One of the proposed mechanisms by which hnRNP-A1 regulates splicing is the intron-exon-intron loop model (Blanchette and Chabot, 1999). In this model, multiple hnRNP-A1 proteins bind to intronic sequences flanking a target exon and multimerize together, thereby forming an intron-exon-intron loop (see Figure 9). The resulting loop is proposed to result in exon skipping. Based on the well-established model of exon definition (Robberson et al., 1990), the mechanism must rely on the inhibition or preclusion of spliceosome protein assembly. Since most splicing and regulatory factors bind within 500 nucleotides upstream or downstream of an exon, we initially set our parameter for this model to require "at least two intronic hnRNP-A1 binding sites within 500 nucleotides, upstream and downstream, of an annotated exon" within the query sequence.

We recognized that, unlike defined proximity to RNA binding site sequences in the previous section, the parameter setting of 500 nucleotides for the loop model would require optimization. To experimentally test (in a bioinformatic manner) whether the parameter value of 500 nt was suitable, we compared the results obtained by setting this

parameter to either 200, 500 or 800 nucleotides and analyzing our reference sequence set of genes with experimentally determined hnRNP-A1 binding sites (Table 3). Both parameter settings of 500 nt and 800 nt correctly identified all known hnRNP-A1 binding sites in the loop model (Table 3, highlighted in red). However, while the parameter of 800 nt did not appear to predict a significant increase in the number of predicted binding sites, it does report binding sites with a significantly broader range, compared to the parameter setting of 500 nt (for example, compare nucleotides 33447-35146 (1700 nucleotide range) versus 33747-34846 (1099 nucleotide range) for the *Src* pre-mRNA). A parameter setting of 200 nucleotides resulted in a clear failure of the program to identify at least all of the known hnRNP-A1 binding sites. Taken together, the data suggest that a setting parameter of approximately 500 nucleotides is appropriate.

3.3 Analysis of reference sequences

Our reference sequence set included genomic DNA sequence for both human and mouse *HNRPA1 (Hnrpa1)*, *SRC (Src)*, *HRAS1 (Hras1)*, *GNAS1 (Gnas1)*, and *FGFR2* (*Fgfr2*) genes, as well as the HIV *tat* gene. As detailed in the introduction, these genes were chosen for our reference sequence set because hnRNP-A1 binding sites in the pre-mRNA transcribed from these genes have all been confirmed experimentally (Blanchette and Chabot, 1999; Bonnal et al., 2005; Del Gatto and Breathnach, 1995; Guil et al., 2003; Hutchison et al., 2002; Pollard et al., 2002; Rooke et al., 2003; Tange et al., 2001; Zhu et al., 2001). Our results show that the current optimized hnRNP-A1 binding site prediction tool correctly predicted all of the validated binding sites described in the literature (Table 4, highlighted in red).

Table 3: hnRNP-A1 binding sites predicted by setting the intron-exon-intron loop parameter to 200 nt, 500 nt or 800 nt[1]

Species	Gene Name	Parameter Setting			Confirmed Binding Site
		200 nt	500 nt	800 nt	
Human	*HNRPA1*	2043, 2081, 2871	300-1314, 2043, 2081, 2867-3730, 3308-3730	2-1377, 2043, 2081, 2867-3807, 3231-4249	2889-3286[2]
	HRAS1	1470, 2371, 2731-2751	1470, 2371, 2731-2751	1470, 2371, 2731-2751	2737[3]
	GNAS1	8356-8362	6763-7807, 7771-8942, 8356	3134-4806, 6463-8107, 7471-9242, 8356, 12905-14602	7268-7331[4]
Mouse	*Src*	44152, 45068	14414-15490, 33747-34846, 39842-40991, 41086-42241, 42644-43823, 44152, 45068	14114-15790, 33447-35146, 39542-41291, 40786-42541, 42344-44123, 44152, 45068	34551[5]

[1] Predicted loops that identify experimentally validated binding sites are indicated in red. Binding sites are indicated as nucleotide position number for the corresponding *Ensembl* genomic sequence for each gene.
[2] (Blanchette M et al, 1999; Chabot B et al, 1997; Hutchison S et al, 2002).
[3] (Guil S et al, 2003).
[4] (Pollard A et al, 2002).
[5] (Rooke N et al, 2003).

Table 4: Predicted hnRNP-A1 binding sites for the reference sequence data set.

Species	Gene	Predicted hnRNP-A1 Binding Site(s)	Validated hnRNP-A1 Binding Site(s)
Human	*SRC*	12975-14124, 14512-15667, 16928-18111, 17379, 17424, 18871	
	HNRPA1	300-1314, 2043, 2081, 2867-3730, 3308-4249	2889-3286[1]
	HRAS1	1470, 2371, 2731-2751	2737[2]
	GNAS1	6763-7807, 7771-8942, 8356	7268-7331[3]
	FGFR2	42331-42950, 55243, 63886, 66227, 66234, 66267, 73596-74236, 75234, 92829-93390, 93342, 95161-95721, 97217, 10996-110517, 105950, 108490	75234[4]
Mouse	*Src*	14414-15490, 33747-34846, 39842-40991, 41086-42241, 42644-43823, 44152, 45068	34551[5]
	Hnrpa1	66-543, 344-1202	
	Hras1	1016-1662, 1346-2314	
	Gnas1	24, 11145-13044, 15201, 15863, 43374, 49361-50433, 51896-52940	
	Fgfr2	None[6]	
Virus	HIV tat	5177-5840, 5394, 7881, 7888, 7894, 7898, 8013	5394, 7881, 7888, 7894, 7898, 8013[7]

[1] (Blanchette M et al, 1999; Chabot B et al, 1997; Hutchison S et al, 2002).
[2] (Guil S et al, 2003).
[3] (Pollard A et al, 2002).
[4] (Del Gatto F et al, 1995).
[5] (Rooke N et al, 2003).
[6] Sequence from 55102 – 105101 nt is not available.
[7] (Zhu J, 2001; Amendt B et al, 1995; Tange T et al, 2001).

In addition, the program predicts a number of additional hnRNP-A1 binding sites (Table 4). While experimental validation of these novel predicted sites is beyond the scope of this thesis, we wished to develop additional tools that might assist the user in evaluating the data reported by this program. The first tool would provide the user with "sequence analysis ready" data in the report to facilitate secondary RNA structure analysis, as described in Section 3.1. The second tool would provide the user with the option to view the data report in a graphical view that shows how the binding sites relate to the intron-exon structure of the gene of interest. Please see Figures 16 and 17 for a complete coverage of the reference set.

4. Discussion

Alternative splicing of pre-mRNA is becoming increasingly recognized as an important regulation of gene expression. Many proto-oncogene products, such as TLS and EWS, have been found to regulate this process and understanding the mechanisms of alternative splicing regulation will lead to a better understanding of how mis-regulation may play a role in the progression of cancer. In this thesis, we developed software to predict hnRNP-A1 binding sites in an effort to develop bioinformatics tools that can be used to identify potential target genes for specific regulatory proteins of pre-mRNA splicing.

hnRNP-A1 has been shown to bind an RNA consensus sequence of UAGGGU/A. Based on random distribution of the four nucleotides, this hexa-nucleotide sequence should be represented in pre-mRNA every 4096 nucleotides [probability of $(1/4)^6$]. A query sequence of 10-15,000 nt, the average length of a gene, would therefore be expected to have two or three binding sites for hnRNP-A1. While this frequency may seem reasonable to use in a search program, consensus sites for RNA binding proteins are highly degenerate and create a frequency of potential binding sites so high that a predictive value based on sequence identity alone becomes uninformative. For example, simply using the 17 possible validated binding sequences (Table 2) predicts more than 60-100 potential binding sites in an average gene sequence. Using experimental approaches to test each individual possible candidate would be costly and inefficient.

In this thesis, we developed a more effective bioinformatics tool to predict hnRNP-A1 binding sites. The program combines the hexa-nucleotide sequences recognized by hnRNP-A1 with the mechanisms by which hnRNP-A1 regulates pre-

mRNA splicing. The result is a significantly reduced number of potential binding sites. The human *HRAS1* gene, for example, is 6432 nt in length with a total of 29 possible hnRNP-A1 binding sequences. However, the combined search in our tool predicts only 3 binding sites, suggesting that more than 86% of false positive sites can be can be eliminated (Figure 15).

Several features of the program contribute to its effectiveness. First, it uses only validated hnRNP-A1 binding site sequences. This ensures the identification of every known hexa-nucleotide binding sequence recognized by hnRNP-A1 while reducing the total number of possible degenerate hexamers that may or may not be biologically relevant. The look-up table itself can be readily updated as new experimental evidence may arise. Second, the program similarly includes all the SR protein binding site sequences so far identified. Third, the program uses exon/intron position data downloaded from the *Ensembl* database, which ensures that the exon/intron positions of the query sequence are correctly annotated. This avoids mistakes that may be caused by using any single exon predicting program, such as GeneScan (Burge and Karlin, 1997) or GenomeScan (Yeh et al., 2001) to mark the intron/exon positions of the query sequence. Fourth, the parameters used to define the relevance of the proximity of hnRNP-A1 sites to the binding sites of other biologically relevant proteins can be adjusted in the program code.

Taken together, these aspects of the program appear to increase the effectiveness of correctly predicting hnRNP-A1 binding sites. Indeed, the results obtained by analyzing known genes regulated by hnRNP-A1 during pre-mRNA splicing correctly identified all experimentally validated hnRNP-A1 binding site in our reference set (Table

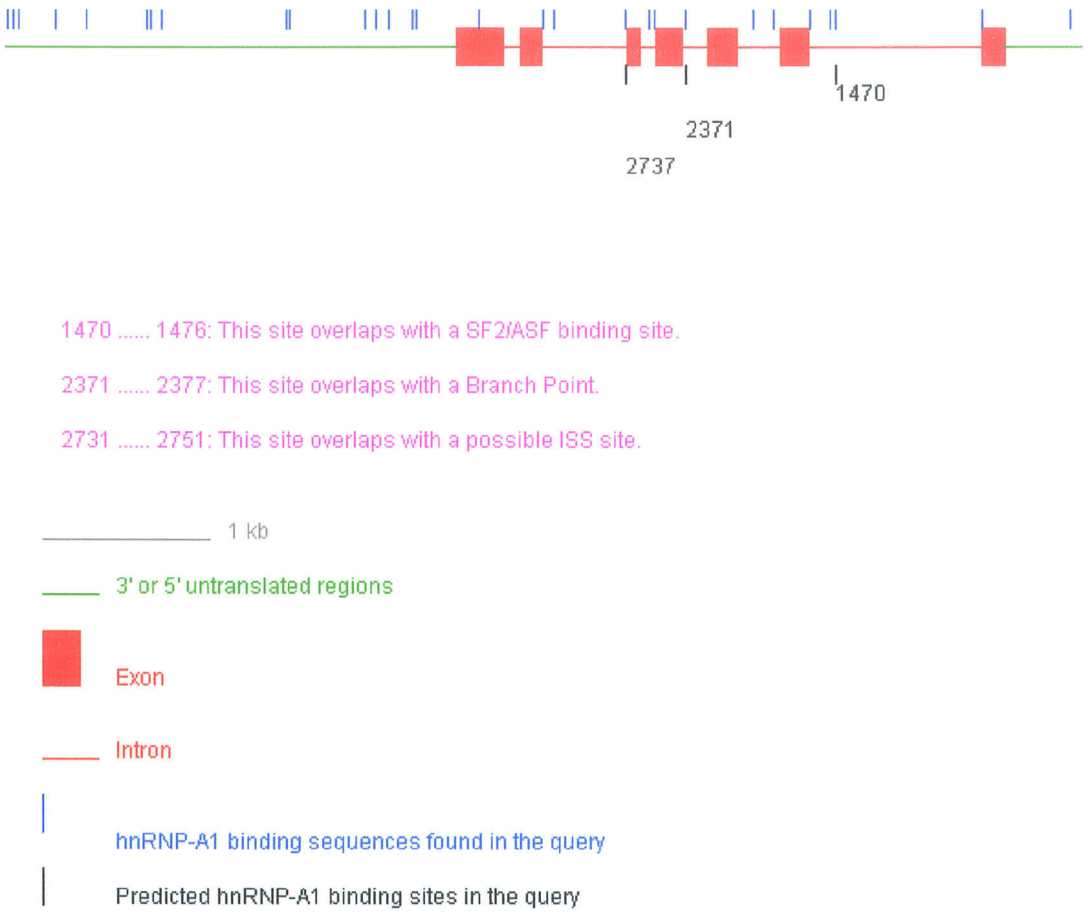

Figure 15: The distribution of hnRNP-A1 binding sequences and predicted hnRNP-A1 binding sites in the human *HRAS1* gene. The locations of hnRNP-A1 binding sequences are marked as blue vertical dashes on the top of the figure, while the predicted hnRNP-A1 binding sites are marked as black vertical dashes on the bottom of the figure. A red square indicates an exon and a red line an intron. A green line marks the 5′ or 3′ untranslated regions. The length of the query sequence can be estimated by the 1 kb grey bar. The figure shows that only three hnRNP-A1 binding sites are predicted and most of the hnRNP-A1 binding sequences are eliminated. The predicted hnRNP-A1 binding site at 2737 has been confirmed experimentally. The possible mechanisms by which hnRNP-A1 predicted sites may be used are also indicated.

4). It is interesting to note that our results also identified additional hnRNP-A1 binding sites. We recognize that the program may have bias for the small reference test set as the hnRNP-A1binding site sequence table itself is based on the experimental data validated from the same reports. This does not, however, preclude the additional binding sites from being truly novel hnRNP-A1 sites; though, the latter can only be determined by additional experimental validation.

To further verify that the identification of novel hnRNP-A1 binding sites has a confidence level higher than random probability, we compared the results with data in the Alternative Splicing Database (ASD) ((Thanaraj et al, 2004); http://www.ebi.ac.uk/asd) for the *HNRPA1* (Hnrpa1) gene to determine whether any novel hnRNP-A1 binding sites correlated with confirmed alternative splice sites. The predicted hnRNP-A1 binding sites were compared with the data from the ASD (Table 4). Only results from genes of human and mouse hnRNP-A1 can be compared with those in the ASD. Three of the novel predicted binding sites are identified in ASD as confirmed alternative splice sites (Table 4, blue highlight; Figure 16 and Figure 17). First, region 2871-3731 was predicted to form an intron-exon-intron loop structure. This region (2889-3286) is confirmed by experiment to be the binding site of hnRNP-A1. Second, hnRNP-A1 is predicted to compete with protein SC35, a member of the SR protein family, to bind to position 2081. Data in the ASD show that regions (1491- 1786) and (2279 - 2791) can be alternatively spliced during pre-mRNA splicing (Figure 16). Thirdly, an intron-exon-intron loop structure was predicted (344-1202) and is within a confirmed alternative splice site (340 - 1242) (Figure 17). Other genes in our reference set were not able to be directly analyzed

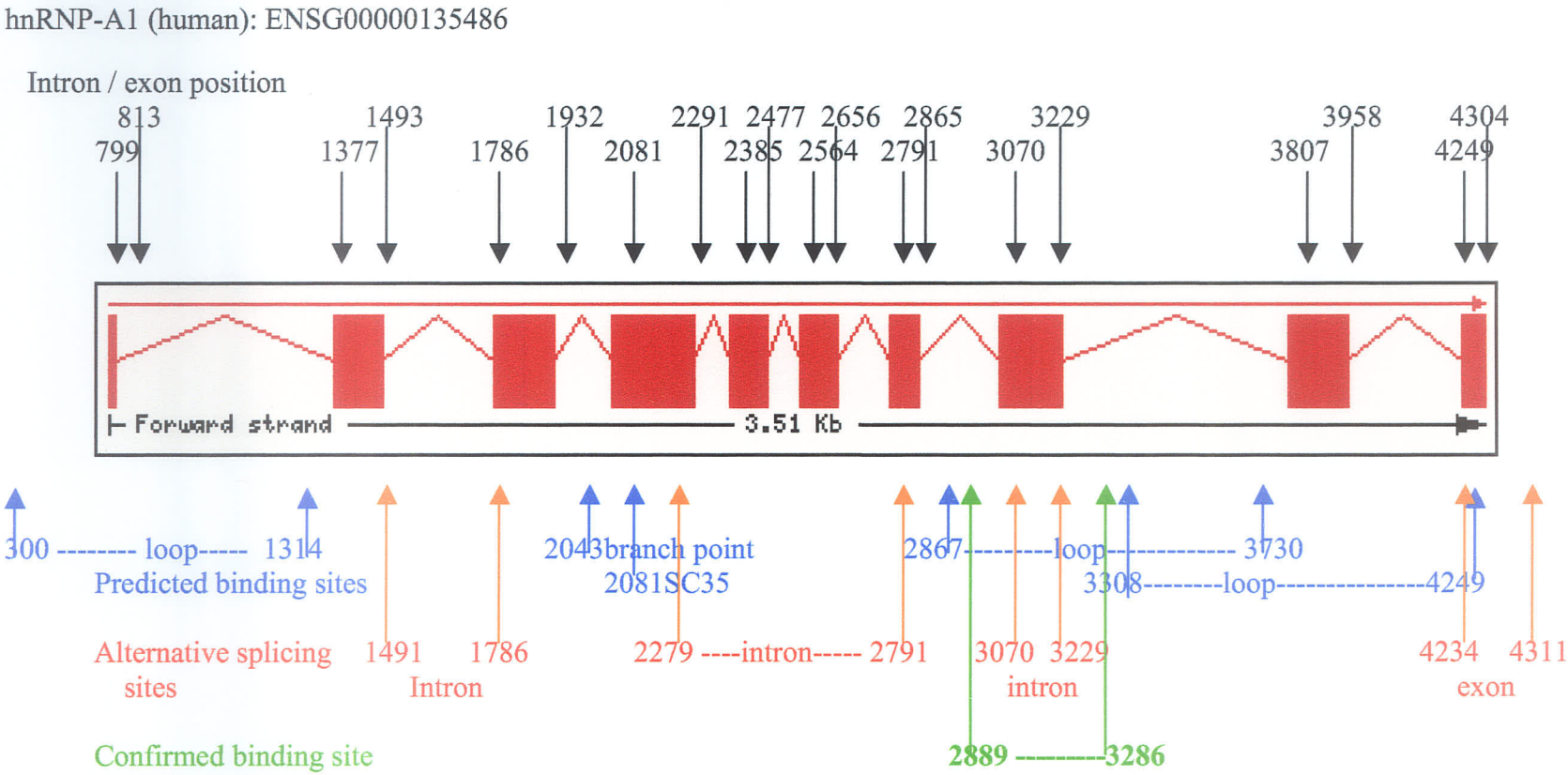

Figure 16: Comparison of the predicted hnRNP-A1 binding sites in human hnRNP-A1 pre-mRNA with the alternative splicing sites found in the Alternative Splicing Database (ASD). Exon/Intron positions are marked with black arrows on the top of the figure, while the confirmed binding sites (in green), predicted binding sites (in blue) and alternative splicing sites found in the ASD (in red) are marked on the bottom of the figure. The figure shows that one experimentally confirmed hnRNP-A1 binding site (2889-3286) is predicted (2867-3730, forming a loop structure) and the other two predicted hnRNP-A1 binding sites (2081 in the SC35 binding site and 2048 in the branchpoint) are close to an alternative splicing region found in the ASD (2279-2791). This may also suggest mechanisms used for the splicing of these sites.

hnRNP-A1 (Mouse): ENSMUSG00000036021

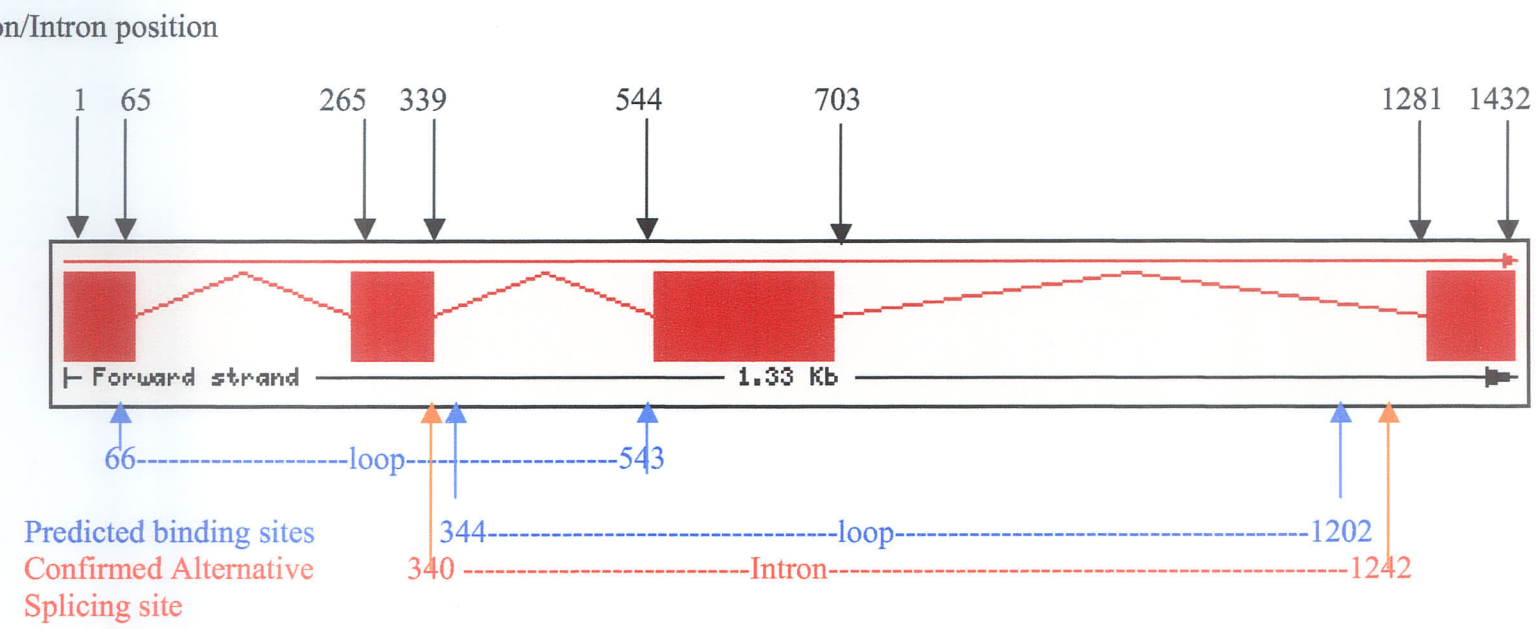

Figure 17: The comparison of the predicted hnRNP-A1 binding sites in mouse hnRNP-A1 pre-mRNA with the alternative splicing sites found in the Alternative Splicing Database (ASD). Exon/Intron positions are marked with black arrows on the top of the figure, while the confirmed binding sites (in green), predicted binding sites (in blue) and alternative splicing sites found in the ASD (in red) are marked on the bottom of the figure. The figure shows that two hnRNP-A1 binding sites are predicted. One predicted hnRNP-A1 binding site (344-1202, forming a loop structure) overlaps an alternative splicing region found in the ASD, which may suggest mechanisms used for the splicing of these sites.

in this manner as they are either not found in the ASD, or differences between exon/intron annotations in the ASD and *Ensembl* databases make them incomparable.

A very recent report (Bonnal et al., 2005) identifying an hnRNP-A1 binding site in the human *FGF2* gene is also encouraging for confidence in the prediction of novel hnRNP-A1 binding sites by our tool. In this report, an hnRNP-A1 binding site in the human *FGF2* gene was deduced to be located between nucleotides 128-144 based on hnRNP-A1 binding in vitro to an FGF-2 RNA fragment spanning nucleotides 1-177, and that hnRNP-A1 binding is greatly decreased in deletion mutants spanning nucleotides 128-144 (Bonnal et al., 2005). Our analysis of *FGF2* pre-mRNA reported two novel hnRNP-A1 binding sites at nucleotides 41-47 and 49115-50218 (note: the sequence they used has an additional 17 nt in the 5′ untranslated region when compared with the sequence exported from the *Ensembl* database). We did not identify any validated hnRNP-A1 binding sequences in nucleotides 128-144, but neither is any possible degenerate binding site apparent. We did, however, identify a predicted hnRNP-A1 binding site in the same intron within 70 nt. It is possible that deletion of nucleotides 128-144 may cause conformational changes that prevent hnRNP-A1 from binding to nucleotides 41-47. Further experiments may be required to test this hypothesis.

The algorithms used in this thesis include the Brute-Force algorithm for hnRNP-A1 binding site sequence searching and the Bubble sort algorithm for array sorting.

For string searching, there are many algorithms available. The most common algorithms include the Brute-Force Algorithm (BF algorithm), the Knuth-Morris-Pratt Algorithm (KMP algorithm), the Boyer-Moore Algorithm (BM algorithm) and Rabin-

Karp Algorithm (RK algorithm). Each of these string searching algorithms has its own characteristics.

In the BF algorithm (Davies et al., 1986), if the pattern P is not matched with the substrings in the text, then the program will slide 1 character distance in the text and start the pattern search again until the matched pattern is found or the end of the text is reached. The code for the BF algorithm is very simple, however, the BF algorithm runs in time O(mn) where m and n are the lengths of the pattern and text, respectively. If both the pattern and text are very large in size, the BF algorithm is not a good choice because of its inefficiency.

The key characteristic of the KMP algorithm (Knuth et al, 1977) is that it uses information about the characters in the text to make a look up table which determines how much to move along the text if a mismatch occurs. The KMP algorithm can greatly reduce the pattern searching time especially when there is no repetition in the pattern. The total KMP run time is O($n+m$) which is significantly more efficient than the BF algorithm.

The BM algorithm (Boyer et al, 1977), on the other hand, works by searching the pattern from right to left but moving the pattern from left to right along the text. When using the BM algorithm, the run time for the algorithm varies between O(n/m) to O(nm). If the first mismatch in the pattern hits a character in the text not in the pattern, the BM algorithm runs in O(n/m). If the last mismatch in the pattern hits a character in the text not in the pattern, the BM run time will be O(mn).

The RK algorithm (Karp et al., 1987) uses a completely different idea for efficient string searching. It first calculates a hash value for the pattern and for each m-character

subsequence in the text. During the string searching, if the hash values between the pattern and an m-character subsequence are unequal, the hash value for the next m-character subsequence will be calculated. However, if the values are equal, the BF algorithm is applied to match the pattern and the m-character subsequence. In most cases, the RK algorithm runs in $O(n+m)$.

The reason that we select the BF algorithm for hnRNP-A1 binding site sequence searching in this thesis is based on the following properties of the problem and of string searching algorithms. First, nucleic acid sequences consist of strings containing only 4 characters (A, G, C and T). Second, both the size of the query sequence (the average size of the gene is 10-15,000 nt) and the sequence search pattern (hnRNP-A1 binding site sequence contains only 6 nt) are quite small. Therefore, using other algorithms is not expected to show much advantage over the BF algorithm. Third, the BF algorithm is the simplest algorithm to implement. Therefore, we decided that using the BF algorithm in this thesis was suitable.

Like string searching, data sorting also has many algorithms available. These include Bubble sorting, Shaker sorting, Shell sorting, Merge sorting etc. The run time for the Bubble sorting algorithm is $O(n^2)$ which is not efficient compared with the other sorting algorithms. Since the number of data items in each array to be sorted in this thesis is very limited, however, the negative asymptotic performance of Bubble sort if not a concern so we chose to use the simple Bubble sorting algorithm.

Finally, when we ran the developed bioinformatics tool for real problems we found that most of the execution time was spent running the BLAST program. The prediction of the hnRNP-A1 binding sites itself took least than 3 seconds, which is an

acceptable waiting time for the tool's users. This confirms that the algorithms selected for use in this thesis are indeed suitable.

4.1 Future directions

We have developed a bioinformatics tool to predict hnRNP-A1 binding sites that is based on known hnRNP-A1 binding site sequences and proximity to binding site sequences for functionally relevant RNA binding proteins based on three hnRNP-A1 mechanisms for the regulation of pre-mRNA splicing (Blanchette and Chabot, 1999; Tange et al., 2001; Zhu et al., 2001). It is important to note that the mechanisms by which hnRNP-A1 regulates pre-mRNA splicing in vivo has not been completely elucidated. Accordingly, we have designed the program to be readily modified as new mechanisms, new hnRNP-A1 binding sequences, new SR proteins and their binding sequences and new ISS(s) and ESS(s) are discovered. The current program has been optimized to correctly identify all known hnRNP-A1 binding sites in a validated reference set. By the same token, the current program may fail to identify novel hnRNP-A1 binding site(s). This may be improved on as more insight into splicing mechanisms is understood, but we freely provide access to our tool in the belief that the scientific community may successfully and advantageously use it in combination with other tools. Indeed, we are actively encouraging the community to beta-test our program and we welcome their comments and suggestions.

Recently, several new bioinformatics tools for predicting exonic splicing enhancers (ESEs) are available (see RESCUE-ESE Web Server at

http://genes.mit.edu/burgelab/rescue-ese/ and ESEfinder at http://rulai.cshl.edu/tools/ESE/index.html). RESCUE (Relative Enhancer and Silencer Classification by Unanimous Enrichment)-ESE uses a computational method and statistical analysis to identify ESEs (Fairbrother et al., 2002). It searches hexameric sequences that have both significantly higher frequency of occurrence in exons than in introns and also significantly higher frequency in exons with weak (non-consensus) splice sites than in exons with strong (consensus) splice sites to identity ESEs candidates. 238 ESEs candidates have been included in its database. ESEfinder is based on the functional SELEX method (Liu et al., 1998) to identify ESEs. A minigene is used that harbors ESE sequences that are required for the efficient splicing of pre-mRNA. The ESE sequence in the minigene is replaced by random hexa-nucleotide sequences. A pool of the pre-mRNA transcribed from the minigene can then be used to interact with individual SR proteins to assay pre-mRNA splicing. If pre-mRNA alternative splicing occurs, the random hexa-nucleotide sequence is PCR amplified and sequenced. The frequencies of the individual nucleotides at each position calculated by the alignment of consensus motif are then used to make a score matrix, which can be used to predict the location of SR-protein-specific ESEs in the query sequence.

Both RESCUE-ESE and ESEfinder are very useful bioinformatics resources for the identification of candidate ESEs and can more precisely identify functional ESEs recognized by SR proteins in the query sequence.

One of the criteria used in this project to predict the hnRNP-A1 binding sites is to check whether SR protein binding sequences overlap by at least one nt with hnRNP-A1 binding site sequences. This may cause false positive hnRNP-A1 binding sites in the

prediction because not every SR protein binding sequence is a functional ESE (SR protein binding site). The integration of the bioinformatics resources such as RESCUE-ESE or ESEfinder with the bioinformatics tool developed in this project may increase the accuracy in hnRNP-A1 binding site prediction as only validated SR protein binding sites would be used for the predicting of hnRNP-A1 binding sites.

cDNA- or oligo-based microarray analysis is another very powerful tool for analyzing gene expression. A single microarray experiment can generate tens of thousands data points of expression information. More recently, the microarray assay has been developed to explore splicing variations, many different types of alternative splicing and genes which undergo alternative splicing have been discovered (Modrek et al, 2002). The locations of the alternative splicing sites in a given gene can easily be identified by comparing the alternatively spliced transcripts with the gene. Microarray assay greatly expands the content of our alternative splicing database.

The alternative splicing database will soon identify most of the genes that can undergo alternative splicing. Comparing the hnRNP-A1 binding sites predicted by my bioinformatics tool with the data in the alternative splicing database would be a very powerful way to eliminate false positive hnRNP-A1 binding sites predicted by this tool. First, if alternative splicing data for a User-specified gene of interest in alternative splicing database indicates the gene does not undergo alternative splicing, then a predicted hnRNP-A1 binding sites is likely to be a false positive. Second, since most alternative splicing regulatory proteins including hnRNP-A1 bind within 500 nt upstream or downstream of the alternative splicing sites, this parameter can also be considered using the alternative splicing database to determine if a predicted hnRNP-A1 binding site

is located out of this range, and therefore, a false positive binding site. Taken together, the hnRNP-A1 binding sites predicted by this bioinformatics tool can be further assessed by comparing the results with those of the alternative splicing database to better determine confidence scores. However, at the current time, most alternative splicing databases only contain very limited alternative splicing information, which greatly limits the significance of the integration of the ASD with this bioinformatics tool.

There are also a number of smaller technical issues where small refinements may enhance performance of the program. For example, the BLAST program is used in this program to identify query sequences and assist the user in selecting the correct *Ensembl* ID. Although the running time of the entire tool varies with the query sequence, between 90-99.9 % of the running time is consumed by BLAST. Run time for the BLAST program depends mostly on the size of the database file, but the length of the query size may also affect the query time. Thus, we use only the first 1,000 nt of the query sequence if its length is more than 1,000 nt. This can considerably reduce the running time of the BLAST program without affecting BLAST results.

Either modifications to the hardware and/or the algorithm itself may also improve the running time of BLAST. Hardware improvements are always desirable, but are not necessarily practical. SSAHA (Sequence Search and Alignment by Hashing Algorithm) (Ning et al., 2001), a fast search method for large DNA databases, is an available program alternative to BLAST. The major difference between SSAHA and BLAST is that, in BLAST, each query sequence is hashed into a k-tuple, while SSAHA hashes the DNA database. This difference is reflected in SSAHA being more efficient in searching large genomic databases for the query nucleotide sequence of genes. When using

SSAHA, the hash table is made only once and the results can be stored in the computer. Then each time a query sequence is submitted, SSAHA can reuse this hash table to find matches to the query sequence. Thus, the running time of the SSAHA is not dependent upon the size of the DNA database, but on the size of the query sequence, which is much smaller than the DNA database. This can dramatically reduce the running time of the sequence searching, especially when the database is very large.

In our own laboratory, we plan to use this program to predict the binding sites of other alternative splicing factors such as TLS. TLS has been shown to be able to regulate pre-mRNA splicing (Calvio et al., 1995; Lerga et al., 2001; Zinszner et al., 1994). The consensus binding sequence has been found to be "GGUG" (Lerga et al., 2001) and the GGUG sequence has been confirmed to bind to a Zn finger (Iko et al., 2004). TLS has also been found to be associated with SR proteins including SC35, SRp75, PTB, SRm160 and TLS-associated serine-arginine (TASR) protein (Meissner et al., 2003; Yang et al., 2000; Yang et al., 1998). While it is clear that more TLS-related information is required, TLS will, to some extent, use similar mechanism(s) as hnRNP-A1 in regulating pre-mRNA splicing. Therefore, it is possible that transposing TLS binding sequence tables and functionally relevant interacting proteins may prove to be a valuable tool to identify candidate target genes for TLS regulation.

5. References

Abovich, N., and Rosbash, M. (1997). Cross-intron bridging interactions in the yeast commitment complex are conserved in mammals. Cell *89*, 403-412.

Alberts, B., Bray, D., Lewis, J., Raff, M., Boberts, K., and Watson, J. (1994). Molecular biology of the cell. Third Edition. Garland Publishing Inc. New York & London. ISBN: 0-8153-1621-6.

Amendt, B. A., Si, Z. H., and Stoltzfus, C. M. (1995). Presence of exon splicing silencers within human immunodeficiency virus type 1 tat exon 2 and tat-rev exon 3: evidence for inhibition mediated by cellular factors. Mol Cell Biol *15*, 4606-4615.

Arning, S., Gruter, P., Bilbe, G., and Kramer, A. (1996). Mammalian splicing factor SF1 is encoded by variant cDNAs and binds to RNA. Rna *2*, 794-810.

Berglund, J. A., Chua, K., Abovich, N., Reed, R., and Rosbash, M. (1997). The splicing factor BBP interacts specifically with the pre-mRNA branchpoint sequence UACUAAC. Cell *89*, 781-787.

Biamonti, G., Buvoli, M., Bassi, M. T., Morandi, C., Cobianchi, F., and Riva, S. (1989). Isolation of an active gene encoding human hnRNP protein A1. Evidence for alternative splicing. J Mol Biol *207*, 491-503.

Black, D. L. (1992). Activation of c-src neuron-specific splicing by an unusual RNA element in vivo and in vitro. Cell *69*, 795-807.

Black, D. L. (2000). Protein diversity from alternative splicing: a challenge for bioinformatics and post-genome biology. Cell *103*, 367-370.

Blanchette, M., and Chabot, B. (1999). Modulation of exon skipping by high-affinity hnRNP A1-binding sites and by intron elements that repress splice site utilization. Embo J *18*, 1939-1952.

Bonnal, S., Pileur, F., Orsini, C., Parker, F., Pujol, F., Prats, A. C., and Vagner, S. (2005). Heterogeneous nuclear ribonucleoprotein A1 is a novel internal ribosome entry site trans-acting factor that modulates alternative initiation of translation of the fibroblast growth factor 2 mRNA. J Biol Chem *280*, 4144-4153.

Bourgeois, C. F., Popielarz, M., Hildwein, G., and Stevenin, J. (1999). Identification of a bidirectional splicing enhancer: differential involvement of SR proteins in 5' or 3' splice site activation. Mol Cell Biol *19*, 7347-7356.

Boyer R., and Moore J. (1977). A fast string searching algoritm. *Communications of the ACM*, 20(10):762-772.

Brudno, M., Gelfand, M. S., Spengler, S., Zorn, M., Dubchak, I., and Conboy, J. G. (2001). Computational analysis of candidate intron regulatory elements for tissue-specific alternative pre-mRNA splicing. Nucleic Acids Res *29*, 2338-2348.

Burd, C. G., and Dreyfuss, G. (1994). RNA binding specificity of hnRNP A1: significance of hnRNP A1 high-affinity binding sites in pre-mRNA splicing. Embo J *13*, 1197-1204.

Burge, C., and Karlin, S. (1997). Prediction of complete gene structures in human genomic DNA. J Mol Biol *268*, 78-94.

Buvoli, M., Cobianchi, F., Bestagno, M. G., Mangiarotti, A., Bassi, M. T., Biamonti, G., and Riva, S. (1990). Alternative splicing in the human gene for the core protein A1 generates another hnRNP protein. Embo J *9*, 1229-1235.

Calvio, C., Neubauer, G., Mann, M., and Lamond, A. I. (1995). Identification of hnRNP P2 as TLS/FUS using electrospray mass spectrometry. Rna *1*, 724-733.

Caputi, M., Mayeda, A., Krainer, A. R., and Zahler, A. M. (1999). hnRNP A/B proteins are required for inhibition of HIV-1 pre-mRNA splicing. Embo J *18*, 4060-4067.

Cartegni, L., Chew, S. L., and Krainer, A. R. (2002). Listening to silence and understanding nonsense: exonic mutations that affect splicing. Nat Rev Genet *3*, 285-298.

Cavaloc, Y., Bourgeois, C. F., Kister, L., and Stevenin, J. (1999). The splicing factors 9G8 and SRp20 transactivate splicing through different and specific enhancers. Rna *5*, 468-483.

Chabot, B., Blanchette, M., Lapierre, I., and La Branche, H. (1997). An intron element modulating 5' splice site selection in the hnRNP A1 pre-mRNA interacts with hnRNP A1. Mol Cell Biol *17*, 1776-1786.

Clark, F., and Thanaraj, T. A. (2002). Categorization and characterization of transcript-confirmed constitutively and alternatively spliced introns and exons from human. Hum Mol Genet *11*, 451-464.

Clinton, J. M., Chansky, H. A., Odell, D. D., Zielinska-Kwiatkowska, A., Hickstein, D. D., and Yang, L. (2002). Characterization and expression of the human gene encoding two translocation liposarcoma protein-associated serine-arginine (TASR) proteins. Gene *284*, 141-147.

Cote, J., Simard, M. J., and Chabot, B. (1999). An element in the 5' common exon of the NCAM alternative splicing unit interacts with SR proteins and modulates 5' splice site selection. Nucleic Acids Res *27*, 2529-2537.

Davies G., and Bowsher S. (1986). Algorithms for pattern matching. *Software--Practice and Experience*, 16(6):575-601.
Del Gatto, F., and Breathnach, R. (1995). Exon and intron sequences, respectively, repress and activate splicing of a fibroblast growth factor receptor 2 alternative exon. Mol Cell Biol *15*, 4825-4834.

Delva, L., Gallais, I., Guillouf, C., Denis, N., Orvain, C., and Moreau-Gachelin, F. (2004). Multiple functional domains of the oncoproteins Spi-1/PU.1 and TLS are involved in their opposite splicing effects in erythroleukemic cells. Oncogene *23*, 4389-4399.

Fairbrother, W. G., Yeh, R. F., Sharp, P. A., and Burge, C. B. (2002). Predictive identification of exonic splicing enhancers in human genes. Science *297*, 1007-1013.

Forch, P., Puig, O., Kedersha, N., Martinez, C., Granneman, S., Seraphin, B., Anderson, P., and Valcarcel, J. (2000). The apoptosis-promoting factor TIA-1 is a regulator of alternative pre-mRNA splicing. Mol Cell *6*, 1089-1098.

Gorlach, M., Wittekind, M., Beckman, R. A., Mueller, L., and Dreyfuss, G. (1992). Interaction of the RNA-binding domain of the hnRNP C proteins with RNA. Embo J *11*, 3289-3295.

Graveley, B. R. (2001). Alternative splicing: increasing diversity in the proteomic world. Trends Genet *17*, 100-107.

Guil, S., Gattoni, R., Carrascal, M., Abian, J., Stevenin, J., and Bach-Elias, M. (2003). Roles of hnRNP A1, SR proteins, and p68 helicase in c-H-ras alternative splicing regulation. Mol Cell Biol *23*, 2927-2941.

Guo, N., and Kawamoto, S. (2000). An intronic downstream enhancer promotes 3' splice site usage of a neural cell-specific exon. J Biol Chem *275*, 33641-33649.

Guth, S., Martinez, C., Gaur, R. K., and Valcarcel, J. (1999). Evidence for substrate-specific requirement of the splicing factor U2AF(35) and for its function after polypyrimidine tract recognition by U2AF(65). Mol Cell Biol *19*, 8263-8271.

Hackl, W., Fischer, U., and Luhrmann, R. (1994). A 69-kD protein that associates reversibly with the Sm core domain of several spliceosomal snRNP species. J Cell Biol *124*, 261-272.

Hallier, M., Lerga, A., Barnache, S., Tavitian, A., and Moreau-Gachelin, F. (1998). The transcription factor Spi-1/PU.1 interacts with the potential splicing factor TLS. J Biol Chem *273*, 4838-4842.

Hedjran, F., Yeakley, J. M., Huh, G. S., Hynes, R. O., and Rosenfeld, M. G. (1997). Control of alternative pre-mRNA splicing by distributed pentameric repeats. Proc Natl Acad Sci U S A *94*, 12343-12347.

Hoffman, D. W., Query, C. C., Golden, B. L., White, S. W., and Keene, J. D. (1991). RNA-binding domain of the A protein component of the U1 small nuclear ribonucleoprotein analyzed by NMR spectroscopy is structurally similar to ribosomal proteins. Proc Natl Acad Sci U S A *88*, 2495-2499.

Hu, G. K., Madore, S. J., Moldover, B., Jatkoe, T., Balaban, D., Thomas, J., and Wang, Y. (2001). Predicting splice variant from DNA chip expression data. Genome Res *11*, 1237-1245.

Huh, G. S., and Hynes, R. O. (1993). Elements regulating an alternatively spliced exon of the rat fibronectin gene. Mol Cell Biol *13*, 5301-5314.

Huh, G. S., and Hynes, R. O. (1994). Regulation of alternative pre-mRNA splicing by a novel repeated hexanucleotide element. Genes Dev *8*, 1561-1574.

Hutchison, S., LeBel, C., Blanchette, M., and Chabot, B. (2002). Distinct sets of adjacent heterogeneous nuclear ribonucleoprotein (hnRNP) A1/A2 binding sites control 5' splice site selection in the hnRNP A1 mRNA precursor. J Biol Chem *277*, 29745-29752.

Iko, Y., Kodama, T. S., Kasai, N., Oyama, T., Morita, E. H., Muto, T., Okumura, M., Fujii, R., Takumi, T., Tate, S., and Morikawa, K. (2004). Domain architectures and characterization of an RNA-binding protein, TLS. J Biol Chem *279*, 44834-44840.

Johnson, J. M., Castle, J., Garrett-Engele, P., Kan, Z., Loerch, P. M., Armour, C. D., Santos, R., Schadt, E. E., Stoughton, R., and Shoemaker, D. D. (2003). Genome-wide survey of human alternative pre-mRNA splicing with exon junction microarrays. Science *302*, 2141-2144.

Kan, Z., Rouchka, E. C., Gish, W. R., and States, D. J. (2001). Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. Genome Res *11*, 889-900.

Karp, RM.; Rabin, MO. (1987). "Efficient randomized pattern-matching algorithms". *IBM Journal of Research and Development* **31** (2), 249-260.

Knuth D, Morris J., and Pratt V. (1977). Fast pattern matching in strings. *SIAM Journal of Computing*, 6(2):323-350.

Lerga, A., Hallier, M., Delva, L., Orvain, C., Gallais, I., Marie, J., and Moreau-Gachelin, F. (2001). Identification of an RNA binding specificity for the potential splicing factor TLS. J Biol Chem *276*, 6807-6816.

Li, X., Zhong, S., and Wong, W. H. (2005). Reliable prediction of transcription factor binding sites by phylogenetic verification. Proc Natl Acad Sci U S A *102*, 16945-16950.

Lipman DJ, Pearson WR. (1985). Rapid and sensitive protein similar search. Science 22; 227(4693): 1435-41

Liu, H. X., Chew, S. L., Cartegni, L., Zhang, M. Q., and Krainer, A. R. (2000). Exonic splicing enhancer motif recognized by human SC35 under splicing conditions. Mol Cell Biol *20*, 1063-1071.

Liu, H. X., Zhang, M., and Krainer, A. R. (1998). Identification of functional exonic splicing enhancer motifs recognized by individual SR proteins. Genes Dev *12*, 1998-2012.

Lou, H., Neugebauer, K. M., Gagel, R. F., and Berget, S. M. (1998). Regulation of alternative polyadenylation by U1 snRNPs and SRp20. Mol Cell Biol *18*, 4977-4985.

Marchand, V., Mereau, A., Jacquenet, S., Thomas, D., Mougin, A., Gattoni, R., Stevenin, J., and Branlant, C. (2002). A Janus splicing regulatory element modulates HIV-1 tat and rev mRNA production by coordination of hnRNP A1 cooperative binding. J Mol Biol *323*, 629-652.

Matlin, A. J., Clark, F., and Smith, C. W. (2005). Understanding alternative splicing: towards a cellular code. Nat Rev Mol Cell Biol *6*, 386-398.

Meissner, M., Lopato, S., Gotzmann, J., Sauermann, G., and Barta, A. (2003). Proto-oncoprotein TLS/FUS is associated to the nuclear matrix and complexed with splicing factors PTB, SRm160, and SR proteins. Exp Cell Res *283*, 184-195.

Modafferi, E. F., and Black, D. L. (1997). A complex intronic splicing enhancer from the c-src pre-mRNA activates inclusion of a heterologous exon. Mol Cell Biol *17*, 6537-6545.

Modrek, B., and Lee, C. (2002). A genomic view of alternative splicing. Nat Genet *30*, 13-19.

Modrek, B., Resch, A., Grasso, C., and Lee, C. (2001). Genome-wide detection of alternative splicing in expressed sequences of human genes. Nucleic Acids Res *29*, 2850-2859.

Mulligan, G. J., Guo, W., Wormsley, S., and Helfman, D. M. (1992). Polypyrimidine tract binding protein interacts with sequences involved in alternative splicing of beta-tropomyosin pre-mRNA. J Biol Chem *267*, 25480-25487.

Ning, Z., Cox, A. J., and Mullikin, J. C. (2001). SSAHA: a fast search method for large DNA databases. Genome Res *11*, 1725-1729.

Pollard, A. J., Krainer, A. R., Robson, S. C., and Europe-Finner, G. N. (2002). Alternative splicing of the adenylyl cyclase stimulatory G-protein G alpha(s) is regulated by SF2/ASF and heterogeneous nuclear ribonucleoprotein A1 (hnRNPA1) and involves the use of an unusual TG 3'-splice Site. J Biol Chem *277*, 15241-15251.

Rapp, T. B., Yang, L., Conrad, E. U., 3rd, Mandahl, N., and Chansky, H. A. (2002). RNA splicing mediated by YB-1 is inhibited by TLS/CHOP in human myxoid liposarcoma cells. J Orthop Res *20*, 723-729.

Reed, R. (1996). Initial splice-site recognition and pairing during pre-mRNA splicing. Curr Opin Genet Dev *6*, 215-220.

Robberson, B. L., Cote, G. J., and Berget, S. M. (1990). Exon definition may facilitate splice site selection in RNAs with multiple exons. Mol Cell Biol *10*, 84-94.

Rooke, N., Markovtsov, V., Cagavi, E., and Black, D. L. (2003). Roles for SR proteins and hnRNP A1 in the regulation of c-src exon N1. Mol Cell Biol *23*, 1874-1884.

Schaal, T. D., and Maniatis, T. (1999). Multiple distinct splicing enhancers in the protein-coding sequences of a constitutively spliced pre-mRNA. Mol Cell Biol *19*, 261-273.

Selvakumar, M., and Helfman, D. M. (1999). Exonic splicing enhancers contribute to the use of both 3' and 5' splice site usage of rat beta-tropomyosin pre-mRNA. Rna *5*, 378-394.

Simard, M. J., and Chabot, B. (2002). SRp30c is a repressor of 3' splice site utilization. Mol Cell Biol *22*, 4001-4010.

Smith, C. W., and Valcarcel, J. (2000). Alternative pre-mRNA splicing: the logic of combinatorial control. Trends Biochem Sci *25*, 381-388.

Southby, J., Gooding, C., and Smith, C. W. (1999). Polypyrimidine tract binding protein functions as a repressor to regulate alternative splicing of alpha-actinin mutally exclusive exons. Mol Cell Biol *19*, 2699-2711.

Tacke, R., and Manley, J. L. (1995). The human splicing factors ASF/SF2 and SC35 possess distinct, functionally significant RNA binding specificities. Embo J *14*, 3540-3551.

Tange, T. O., Damgaard, C. K., Guth, S., Valcarcel, J., and Kjems, J. (2001). The hnRNP A1 protein regulates HIV-1 tat splicing via a novel intron silencer element. Embo J *20*, 5748-5758.

Thanaraj, T., Stamm, S., Clark, F., Riethoven, J., Le Texier V., Muilu, J. (2004). ASD: the alternative splicing database. Nucleic Acids Res. 32, D64-D69.

Tuerk, C., Gold, L. (1990). Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase. Science 249, 505-510.

Varani, G. (1995). Exceptionally stable nucleic acid hairpins. Annu Rev Biophys Biomol Struct *24*, 379-404.

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., Smith, H. O., Yandell, M., Evans, C. A., Holt, R. A., *et al.* (2001). The sequence of the human genome. Science *291*, 1304-1351.

Wittekind, M., Gorlach, M., Friedrichs, M., Dreyfuss, G., and Mueller, L. (1992). 1H, 13C, and 15N NMR assignments and global folding pattern of the RNA-binding domain of the human hnRNP C proteins. Biochemistry *31*, 6254-6265.

Wu, J. Y., and Maniatis, T. (1993). Specific interactions between proteins implicated in splice site selection and regulated alternative splicing. Cell *75*, 1061-1070.

Wu, S., Romfo, C. M., Nilsen, T. W., and Green, M. R. (1999). Functional recognition of the 3' splice site AG by the splicing factor U2AF35. Nature *402*, 832-835.
Yang, L., Embree, L. J., and Hickstein, D. D. (2000). TLS-ERG leukemia fusion protein inhibits RNA splicing mediated by serine-arginine proteins. Mol Cell Biol *20*, 3345-3354.

Yang, L., Embree, L. J., Tsai, S., and Hickstein, D. D. (1998). Oncoprotein TLS interacts with serine-arginine proteins involved in RNA splicing. J Biol Chem *273*, 27761-27764.

Yeakley, J. M., Fan, J. B., Doucet, D., Luo, L., Wickham, E., Ye, Z., Chee, M. S., and Fu, X. D. (2002). Profiling alternative splicing on fiber-optic arrays. Nat Biotechnol *20*, 353-358.

Yeh, R. F., Lim, L. P., and Burge, C. B. (2001). Computational inference of homologous gene structures in the human genome. Genome Res *11*, 803-816.

Zhu, J., Mayeda, A., and Krainer, A. R. (2001). Exon identity established through differential antagonism between exonic splicing silencer-bound hnRNP A1 and enhancer-bound SR proteins. Mol Cell *8*, 1351-1361.

Zinszner, H., Albalat, R., and Ron, D. (1994). A novel effector domain from the RNA-binding protein TLS or EWS is required for oncogenic transformation by CHOP. Genes Dev *8*, 2513-2526.

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. Nucleic Acids Res *31*, 3406-3415.

6. Appendix: Programming code created in this project

index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<frameset rows="*" cols="120,*" framespacing="0" frameborder="NO" border="0">
  <frame src="FirstFrame.php" name="leftFrame" scrolling="NO" noresize>
  <frame src="project.php" name="showframe">
</frameset>
<noframes><body>

</body></noframes>
</html>
```

FirstFrame.php

```
<html>
  <body background = "background.gif">
  <img src="CancerCare240.JPG" width="150" height="100"><br><br>
  <a href="http://www.umanitoba.ca" target="showframe">University of Manitoba </a><br><br>
  <a href="http://www.umanitoba.ca/institutes/manitoba_institute_cell_biology" target="showframe">
        Manitoba Institute of Cell Biology</a><br><br>
  <a href="http://www.escells.ca" target="showframe">Mammalian Functional Genomic Centre</a><br><br>
  <a href="bioinformaticTool.php" target="showframe">Useful Bioinformatics Tools</a><br><br>
  <a href="http://140.193.242.11/project.php" target="showframe">Home</a><br><br>

  </body>
</html>
```

project.php

```
<html><head></head>
  <body background="background.gif">
  <form action="action.php" method="post">

  <h1 align="center" style="font-size:150%"><font color="blue">hnRNP-A1 BINDING SITE(S) PREDICTION TOOL</font></h1>
  <img src="titlebar.gif" align="middle" width="800" height="5"><br><br>
  <p>This software is still in its <font color="red">TEST STAGE</font>.
       We welcome people to use it and feed back your comments and suggestion.</p>

  <p>Two steps to predict hnRNP-A1 binding sites in your query sequence:</p>
  <p>(1) Paste your query sequence in FASTA format or select a file containing
       your query sequence in FASTA format, and click Submit. A BLAST results
       will be popped out in a new window.</p>
  <p>(2) Select ENSEMBL ID of your query sequence and click Continue. The results
       of hnRNP-A1 binding site prediction will be shown.</p><br><br>
  <p> <h4 align="left">Paste your sequence in FASTA format here</h4> </p>

  <textarea rows="10" cols="70" name="blast"> </textarea> <br><br>

  or Select your file:<br>

  <input type="file" value="Browse" name = "userfile"> <br><br>

  Select Speices:<br>
  <input type="radio" name="species" value="Human"> Human
  <input type="radio" name="species" value="Mouse"> Mouse

  <br><br>


  <input type="submit" value="Submit">

  <input type="reset" value="Reset">

  </form>
```

```
<br><br>

<img align="middle" src="titlebar.gif" width="800" height="5"><br><br>
Contact:<br>
Dr. Geoff Hicks</a><br>
Manitoba Institute of Cell Biology 675 McDermot Ave. Rm. ON5029 Winnipeg, MB R3E 0V9<br>
Tel: (204) 787-2133, Lab: (204) 787-2167 Fax: (204) 787-2190
<a href="mailto:hicksgg@cc.umanitoba.ca">hicksgg@cc.umanitoba.ca</a>.<br><br>

This website is created and maintained by Dr. Yanglong Mou</a>.
You can also send email to <a href="mailto:mouy@cc.umanitoba.ca">mouy@cc.umanitoba.ca</a>.<br><br>

This website was created in July 2005.<br>

</body>
</html>
```

action.php

```
<html><head></head><body background="background.gif">
 <h1 align = "center" style="font-size:150%"> BLAST results for your query sequence </h1>


  <?php


     while (list($key, $value) = each($HTTP_POST_VARS))
     {
            //print "Key: $key: Value: $value<br />";

            if ($key == 'blast')
        {
                    $blast = $value;
            }
            if ($key == 'userfile')
        {
                    $file = $value;
            }
            if ($key == 'species')
        {
                    $Species = $value;
            }
     }


     $fileName = fopen ("tempFile.txt", "w");
     if (!$fileName)
     {
         echo "<p>Unable to open remote file for writing.\n";
         exit;
     }

     if ($file != "")
     {
```

```
        fwrite ($fileName, "1\n");
        fwrite ($fileName, $file);
    }
    if ($file == "" && $blast != "")
    {
        fwrite ($fileName, "0\n");
        fwrite ($fileName, $blast);
    }
    if (($file == "" && $blast == " ")||($Species == ""))
    {
        echo "<h2>ERROR!!!</h2>";
        echo "<h3>You haven't pasted your query sequence or selected your query sequence file or selected species!\n</h3>";
        exit;
    }
    fclose ($fileName);



    $callReadQuerySeq = shell_exec('java ReadQuerySeq');
    //echo $callReadQuerySeq;


     if ($Species=="Human")
     {
        exec("c:\ResearchProj\BLAST\blastall.exe -p blastn -d c:\ResearchProj\BLAST\EnsGeneSeq -i subSequence.txt -o
c:\ResearchProj\Files\blastResult.txt");
     }
     elseif ($Species=="Mouse")
     {
            exec("c:\ResearchProj\BLAST\blastall.exe -p blastn -d c:\ResearchProj\BLAST\EnsMouseGeneSeq -i subSequence.txt -o
c:\ResearchProj\Files\blastResult.txt");
     }

 ?>

 <a href="page.php" target="_blank">To view a full page, click here!</a>
 <textarea rows="15" cols="90" name="blastFinal" style="font-family: monospace; font-size: 12px;" READONLY >
```

```
<?php

 $file = fopen ("c:\ResearchProj\Files\blastResult.txt", "r");
 if (!$file)
 {
    echo "<p>Unable to open remote file.\n";
    exit;
 }

 $lineNumber = 0;
 $idArray = array(0 => ' ', 1 => ' ', 2 => ' ', 3 => ' ', 4 => ' ',
                  5 => ' ', 6 => ' ', 7 => ' ', 8 => ' ', 9 => ' ');


 while (!feof ($file))
 {
    $line = fgets ($file, 1024);
    print "$line";


    if (($line{0}=='E')&&($line{1}=='N')&&($line{2}=='S')&&($lineNumber < 10))
    {
       $idArray[$lineNumber] = $line;
       $lineNumber = $lineNumber + 1;
    }

 }
 fclose($file);
?>

</textarea>

<form name="input" action="FinalResult.php" method="get">
Select Ensembl ID of your query sequence:  

<select name="EnID">
```

```
 <option value="<?php print "$idArray[0]"; ?>"><?php print "$idArray[0]"; ?>
 <option value="<?php print "$idArray[1]"; ?>"><?php print "$idArray[1]"; ?>
 <option value="<?php print "$idArray[2]"; ?>"><?php print "$idArray[2]"; ?>
 <option value="<?php print "$idArray[3]"; ?>"><?php print "$idArray[3]"; ?>
 <option value="<?php print "$idArray[4]"; ?>"><?php print "$idArray[4]"; ?>
 <option value="<?php print "$idArray[5]"; ?>"><?php print "$idArray[5]"; ?>
 <option value="<?php print "$idArray[6]"; ?>"><?php print "$idArray[6]"; ?>
 <option value="<?php print "$idArray[7]"; ?>"><?php print "$idArray[7]"; ?>
 <option value="<?php print "$idArray[8]"; ?>"><?php print "$idArray[8]"; ?>
 <option value="<?php print "$idArray[9]"; ?>"><?php print "$idArray[9]"; ?>
 </select>
 <input type="submit" value="Continue">


 </form>
</body>
</html>
```

page.php

```
<html><head></head><body background="background.gif">

 <h1 align = "center" style="font-size:150%"> BLAST results for your query sequence </h1>

 <textarea rows="15" cols="90" name="blastFinal" style="font-family: monospace; font-size: 13px;" >


 <?php

  $file = fopen ("c:\ResearchProj\Files\blastResult.txt", "r");
  if (!$file)
  {
     echo "<p>Unable to open remote file.\n";
     exit;
  }

  $lineNumber = 0;
  $idArray = array(0 => ' ', 1 => ' ', 2 => ' ', 3 => ' ', 4 => ' ',
                5 => ' ', 6 => ' ', 7 => ' ', 8 => ' ', 9 => ' ');


  while (!feof ($file))
  {
     $line = fgets ($file, 1024);
     print "$line";


     if (($line{0}=='E')&&($line{1}=='N')&&($line{2}=='S')&&($lineNumber < 10))
     {
        $idArray[$lineNumber] = $line;
        $lineNumber = $lineNumber + 1;
     }

  }
  fclose($file);
```

```
?>

</textarea>

<form name="input" action="FinalResult2.php" method="get">
Select Ensembl ID of your query sequence:  

<select name="EnID">
<option value="<?php print "$idArray[0]"; ?>"><?php print "$idArray[0]"; ?>
<option value="<?php print "$idArray[1]"; ?>"><?php print "$idArray[1]"; ?>
<option value="<?php print "$idArray[2]"; ?>"><?php print "$idArray[2]"; ?>
<option value="<?php print "$idArray[3]"; ?>"><?php print "$idArray[3]"; ?>
<option value="<?php print "$idArray[4]"; ?>"><?php print "$idArray[4]"; ?>
<option value="<?php print "$idArray[5]"; ?>"><?php print "$idArray[5]"; ?>
<option value="<?php print "$idArray[6]"; ?>"><?php print "$idArray[6]"; ?>
<option value="<?php print "$idArray[7]"; ?>"><?php print "$idArray[7]"; ?>
<option value="<?php print "$idArray[8]"; ?>"><?php print "$idArray[8]"; ?>
<option value="<?php print "$idArray[9]"; ?>"><?php print "$idArray[9]"; ?>
</select>
<input type="submit" value="Continue">
</form>
</body>
</html>
```

SecondPage.php

```
<html><body background="background.gif">

 <h3 align="center">Results for hnRNP-A1 binding site(s) prediction</h3>

 <textArea rows="18" cols="90" name="resultFinal" style="font-family: monospace; font-size: 13px;" READONLY>

 <?php

    $fileRead = fopen ("c:\ResearchProj\Files\Result.txt", "r");
    if (!$fileRead)
    {
       echo "<p>Unable to open remote file.\n";
       exit;
    }

    while (!feof ($fileRead))
    {
       $line = fgets ($fileRead, 1024);

       print "$line";
    }

    fclose($fileRead);

 ?>

 </textArea><br>

 <a href="http://www.bioinfo.rpi.edu/applications/mfold/old/rna/" target="_blank">
 Click to check RNA Secondary Structure</a>         
       
 <a href="picture.php" target="_blank">Click to view A1 binding site distribution!</a>
</body>
</html>
```

FinalResult.php

```
<html><body background="background.gif">

 <h2 align="center" style="font-size:150%">Results for hnRNP-A1 binding site(s) prediction</h2>


 <?php

 $SeqID = $_GET['EnID'];

 if ($SeqID == "")
 {
    echo "<h2>You haven't entered Ensembl ID of your query sequence!!!</h2>";
    exit;
 }

 $openfile = fopen ("tempFile.txt", "w");
 if (!$openfile)
 {
    echo "<p>Unable to open remote file for writing.\n";
    exit;
 }
 fwrite ($openfile, $SeqID);


 $callJava = shell_exec('Java prediction');

 if ($callJava == "")
 {

    $fileRead = fopen ("c:\ResearchProj\Files\Result.txt", "r");
    if (!$fileRead)
    {
       echo "<p>Unable to open remote file.\n";
       exit;
    }
```

```
?>

<a href="SecondPage.php" target="_blank">To view full page, click here!</a>
<textArea rows="17" cols="90" name="resultFinal" style="font-family: monospace; font-size: 13px;" READONLY >

<?php

  while (!feof ($fileRead))
  {
     $line = fgets ($fileRead, 1024);
     if (($line{1} == ')') || ($line{2} == ')'))
     {
          print "$line";
     }
     else
     {
         print "$line";
     }
  }

  fclose($fileRead);
}
else
{
    print "<h2>$callJava</h2>";
    exit;
}
?>
</textArea>
<a href="http://www.bioinfo.rpi.edu/applications/mfold/old/rna/" target="_blank">
Click to check RNA Secondary Structure</a>         
       
<a href="picture.php" target="_blank">Click to view A1 binding site distribution!</a>
</body>
</html>
```

FinalResult2.php

```
<html><body background="background.gif">

 <h2 align="center" style="font-size:150%">Results for hnRNP-A1 binding site(s) prediction</h2>


 <?php

 $SeqID = $_GET['EnID'];

 if ($SeqID == "")
 {
    echo "<h2>You haven't entered Ensembl ID of your query sequence!!!</h2>";
    exit;
 }

 $openfile = fopen ("tempFile.txt", "w");
 if (!$openfile)
 {
    echo "<p>Unable to open remote file for writing.\n";
    exit;
 }
 fwrite ($openfile, $SeqID);


 $callJava = shell_exec('Java prediction');

 if ($callJava == "")
 {

    $fileRead = fopen ("c:\ResearchProj\Files\Result.txt", "r");
    if (!$fileRead)
    {
       echo "<p>Unable to open remote file.\n";
       exit;
    }
```

```
?>


<textArea rows="17" cols="90" name="resultFinal" style="font-family: monospace; font-size: 13px;" READONLY >

<?php

  while (!feof ($fileRead))
  {
     $line = fgets ($fileRead, 1024);

     print "$line";
  }

  fclose($fileRead);
}
else
{
    print "<h2>$callJava</h2>";
    exit;
}

?>
</textArea><br>

<a href="http://www.bioinfo.rpi.edu/applications/mfold/old/rna/" target="_blank">
Click to check RNA Secondary Structure</a>         
      
<a href="picture.php" target="_blank">Click to view A1 binding site distribution!</a>
</body>
</html>
```

bioinformaticsTool.php

```
<html><head></head><body background="background.gif">

  <h2 align = "center" style="font-size:150%"> Useful Bioinformatic Tools </h2>

  <h3>Useful Software:</h3>
  <ul>
  <li>DNA/RNA or Protein Sequence Alignment: <a href="http://www.ncbi.nlm.nih.gov/BLAST/" target="_blank">BLAST in NCBI</a><br>
  <li>DNA/RNA or Protein Sequence Alignment: <a href="http://www.ensembl.org/Multi/blastview" target="_blank">SSAHA and BLAST in
Ensembl</a><br>
  <li>Multiple Sequence Alignment: <a href="http://www.ebi.ac.uk/clustalw/" target="_blank">ClustalW</a><br>
  <li>Primer Design in PCR: <a href="http://frodo.wi.mit.edu/cgi-bin/primer3/primer3_www.cgi" target="_blank">Primer3 </a><br>
  <li>Restriction Mapping Nucleotide Sequences: <a href="http://users.unimi.it/~camelot/tools/cut2.html" target="_blank">Webcutter
</a><br>
  <li>Gene-finding in Eukaryotes: <a href="http://genome.ucsc.edu/index.html" target="_blank">UCSC</a><br>
  <li>RNA Secondary Structure Prediction: <a href="http://www.bioinfo.rpi.edu/applications/mfold/old/rna/form1.cgi"
target="_blank">MFOLD</a><br>
  <li>Protein Fold Recognition: <a href="http://www.bmm.icnet.uk/people/rob/CCP11BBS/foldrec.html" target="_blank">Fold Recognition
Methods and Links</a><br>
  <li>Make a Pictogram for Nuclear Acid Sequence: <a href="http://genes.mit.edu/pictogram.html" target="_blank">Make a
Pictogram</a><br>

  </ul>
  <h3>Useful Databases:</h3>
  <ol>
  <li><h4>Nucleotide Sequence Databases:</h4>
     <ul>
       <li>Annotated Collection of Nucleotide and Protein Sequences: <a href="http://www.ncbi.nlm.nih.gov/"
target="_blank">GenBank</a><br>
       <li>Annotated Collection of Nucleotide and Protein Sequences: <a href="http://www.ebi.ac.uk/embl.html" target="_blank">EMBL
Nucleotide Sequence Database</a><br>
       <li>Unified Clusters of ESTs and Full-length mRNA Sequences: <a href="http://www.ncbi.nlm.nih.gov/UniGene/"
target="_blank">UniGene</a><br>
       <li>Characterization and Classification of Nucleic Acid Vectors: <a href="http://genome-www2.stanford.edu/vectordb/"
target="_blank">VectorDB</a><br>
       <li>EBI's Alternative Splicing Database: <a href="http://www.ebi.ac.uk/asd" target="_blank">ASD</a><br>
```

```
      <li>Exon-Intron database: <a href="http://mcb.harvard.edu/gilbert/EID/" target="_blank">EID</a><br>
      <li>Binding Sites for E.coli DNA-binding Proteins: <a href="http://arep.med.harvard.edu/dpinteract" target="_blank">DPInteract</a><br>
      <li>Transcription Factors and Binding Sites: <a href="http://transfac.gbf.de/TRANSFAC/index.html"
target="_blank">TRANSFAC</a><br>
      <li>All Complete rRNA Sequences: <a href="http://www.psb.ugent.be/rRNA/" target="_blank">European rRNA Database</a><br>
      <li>Genomic tRNA Database: <a href="http://rna.wustl.edu/GtRDB" target="_blank">GtRDB</a><br>
    </ul><br><br>
  <li><h4>Protein Sequence Databases:</h4>
    <ul>
      <li>Sequences of Proteins with Experimentally Verified Function: <a href="http://www.cmbi.kun.nl/EXProt/"
target="_blank">EXProt</a><br>
      <li>Protein Information Resource:&bnsp<a href="http://pir.georgetown.edu/" target="_blank">PIR</a><br>
      <li>Curated Protein Sequence Database: <a href="http://www.expasy.org/sprot" target="_blank">Swiss-Prot</a><br>
      <li>Protein-protein Interaction Database: <a href="http://www.bind.ca/Action" target="_blank">BIND</a><br>
      <li>Database of Protein Subcellular Localization: <a href="http://www.bioinfo.tsinghua.edu.cn/dbsubloc.html"
target="_blank">DBSubLoc</a><br>
      <li>O- and C-linked Glycosylation Sites in Proteins: <a href="http://www.cbs.dtu.dk/databases/OGLYCBASE/" target="_blank">O-
GlycBase</a><br>
      <li>Protein Phosphorylation Sites: <a href="http://www.cbs.dtu.dk/databases/PhosphoBase/" target="_blank">PhosphoBase</a><br>
      <li>Conserved Domain Database: <a href="http://www.ncbi.nlm.nih.gov/Structure/cdd/cdd.shtml" target="_blank">CDD</a><br>
      <li>A Database of Protein domains and motifs: <a href="http://hits.isb-sib.ch/" target="_blank">Hits</a><br>
      <li>Integrated Resource of Protein Families, Domains and Functional Sites: <a href="http://www.ebi.ac.uk/interpro"
target="_blank">InterPro</a><br>
      <li>G-protein-coupled Receptors Database: <a href="http://www.gpcr.org/7tm/" target="_blank">GPCRDB</a><br>

    </ul><br><br>
  <li><h4>Structure Databases:</h4>
    <ul>
      <li>Amino Acid-nucleotide interaction database:<a href="http://aant.icmb.utexas.edu/" target="_blank">AANT</a><br>
      <li>Protein Domain Structures Database:<a href="http://www.biochem.ucl.ac.uk/bsm/cath_new" target="_blank">CATH</a><br>
      <li>Protein Fold Predcition from Genome Sequence:<a href="http://spock.genes.nig.ac.jp/~genome/" target="_blank">GTOP</a><br>
      <li>Integrated Sequence-structure Database:<a href="http://www.protein.bio.msu.su/issd" target="_blank">ISSD</a><br>
      <li>NCBI's Database of 3D Structures:<a href="http://www.ncbi.nlm.nih.gov/Structure" target="_blank">MMDB</a><br>

    </ul><br><br>

  <li><h4>Metabolic Enzymes and Pathways; Signaling Pathways:</h4>
```

```
    <ul>
      <li>Enzyme Nomenclature and Properties:<a href="http://www.expasy.org/enzyme" target="_blank">ENZYME</a><br>
      <li>Enzyme Names and Properties:<a href="http://www.brenda.uni-koeln.de" target="_blank">BRENDA</a><br>
      <li>Metabolic Pathways and Enzymes from Various Organisms:<a href="http://www.metacyc.org" target="_blank">MetaCyc</a><br>
      <li>Biochemical Pathways, Compounds and Metabolism:<a href="http://www.ncgr.org/pathdb" target="_blank">PathDB</a><br>
      <li>Biomolecular Interaction Network Database:<a href="http://www.bind.ca" target="_blank">BIND</a><br>
      <li>Signaling Pathway Networks:<a href="http://www.amaze.ulb.ac.be/" target="_blank">aMAZE</a><br>
      <li>Protein-protein Interaction Data:<a href="http://www.ebi.ac.uk/intact" target="_blank">IntAct Project</a><br>
      <li>Putative Protein Domain Interactions:<a href="http://interdom.lit.org.sg" target="_blank">InterDom</a><br>
      <li>Signal Transductions Classification Database:<a href="http://www.techfak.uni-bielefeld.de/~mchen/STCDB"
target="_blank">STCDB</a><br>

    </ul><br><br>

  <li><h4>Human Genes and Diseases:</h4>
    <ul>
      <li>A General Guide on Human Hereditary Diseases:<a href="http://ghr.nlm.nih.gov/" target="_blank">Genetic Home Reference</a><br>
      <li>Online Mendelian Inheritance in Animals:<a href="http://www.angis.org.au/omia" target="_blank">OMIA</a><br>
      <li>Online Mendelian Ingeritance in Man:<a href="http://www.ncbi.nlm.nih.gov/Omim/" target="_blank">OMIM</a><br>
      <li>European Mutatnt Mice Pathology Database:<a href="http://www.pathbase.net/" target="_blank">PathBase</a><br>
      <li>Compilation of Protein Mutant Data:<a href="http://pmd.ddbj.nig.ac.jp/" target="_blank">PMD</a><br>
      <li>Allele Frequencies and DNA Polymorphisms:<a href="http://alfred.med.yale.edu/" target="_blank">ALFRED</a><br>
      <li>A Compilation of Human Mutation Databases:<a href="http://www.hgvs.org/" target="_blank">HGVS Databases</a><br>
      <li>Human Gene Mutation Database:<a href="http://www.hgmd.org/" target="_blank>HGMD</a><br>
      <li>Cancer Related Genes, Chromosomal Abnormalities in Oncology and Haematology:<a
href="http://www.infobiogen.fr/services/chromcancer/" target="_blank">
          Altas of Genetics and Cytogenetics in Oncology and Haematology</a><br>
      <li>Cancer Gene Expression Database:<a href="http:://love2.aist-nara.ac.jp/CGED" target="_blank">CGED</a><br>
      <li>Mouse Tumor Biology Database:<a href="http://tumor.informatics.jax.org/" target="_blank">MTB</a><br>
      <li>Cellular, Molecular and Biological Data about Genes Involved in Various Cancers:<a href="http://www.tumor-gene.org/tgdf.html"
target="_blank">
          Tumor Gene Family Databases</a><br>
    </ul><br><br>
  <li><h4>Microarray Data and othe Gene Expression Databases:</h4>
    <ul>
      <li>Public Collection of Microarray Gene Expression Data:<a href="http://www.ebi.ac.uk/arrayexpress"
target="_blank">ArrayExpress</a><br>
```

```
      <li>Human and Mouse Gene Expression Data:<a href="http://bodymap.ims.u-tokyo.ac.jp/" target="_blank">BodyMap</a><br>
      <li>Human Genes Expression Profiles in Healthy Tissues:<a href="http://genecards.weizmann.ac.il/genenote/"
target="_blank">GeneNote</a><br>
      <li>Gene Expression Patterns in the Mouse:<a href="http://www.genepaint.org/Frameset.html" target="_blank">GenePaint</a><br>
      <li>Expression Patterns in an Embryonic Stem Library of Gene Trap Insertions:<a href="http://www.cmhd.ca/sub/genetrap.asp"
target="_blank">GeneTrap</a><br>
      <li>Mouse Embryonic Stem Cell Library with Defined Mutaion:<a href="http://www.escells.ca" target="_blank">Mammalian Functional
Genomic Centre</a><br>
      <li>Mouse Gene Expression Database:<a href="http://www.informatics.jax.org/menus/expression_menu.shtml"
target="_blank">GXD</a><br>
      <li>Reference Database for Human Gene Expression Analysis:<a href="http://www.lsbm.org/db/index_e.html"
target="_blank">RefExA</a><br>
     </ul><br><br>
  <li><h4>Proteomics Resources:</h4>
     <ul>
      <li>2D Gel Electrophoresis Patterns of Proteins from Complete Microbial Genomes:<a href="http://gelbank.anl.gov/"
target="_blank">GelBank</a><br>
      <li>Predictions for Entire Proteomes - Summarized Analyses of Protein Sequences:<a href="http://cubic.bioc.columbia.edu/pep/"
target="_blank">PEP</a><br>
      <li>Functional Classification of Proteins in Whole Genomes:<a href="http://www.ebi.ac.uk/proteome/" target="_blank">Proteome Analysis
Database</a><br>
      <li>Annotated 2D Gel Electrophoresis Database:<a href="http://www.expasy.org/ch2d/" target="_blank">SWISS-2DPAGE</a><br>
     </ul>
  </ol>
 </body>
</html>
```

picture.php

```
<HTML><HEAD></HEAD><BODY>
<SCRIPT>
alert("Screen Dimension\n" +
  "  width:" +
  java.awt.Toolkit.getDefaultToolkit().getScreenSize().width +
  " height:" +
  java.awt.Toolkit.getDefaultToolkit().getScreenSize().height);
</SCRIPT>

<APPLET CODE="Figure.class"
     NAME="myApplet"
       HEIGHT=800 WIDTH=1000>
</APPLET>

</BODY></HTML>
```

prediction.java

```
/*Bioinformatics tool for hnRNP-A1 binding site prediction
  Web site address: http://140.193.242.11/
  The code is written by Dr. Yanglong Mou
  Contact information: tel: (204)-787-4584*/

/*The basic idea is to find out A1 binding site sequences and
  intron/exon position of the query sequence than predicting
  possible A1 binding sites*/

/*This program is launched after the BLAST when user selects
  the Ensembl ID of the query*/

import java.io.*;

public class prediction
{
        public static void main(String[] args)
        {
                String SeqTitle = "";           //the first line of the query sequence in FASTA form
                String fullSeq = "";                                    //the query sequence
                int seqLength = 0;                                              //the length of the query sequence
                int species = 0;                                        //the species of which the query sequence is derived

                /*call readFullSeq function in FileRead class
                 to get the first line of the query sequence, full query seqence,
                and the length of the query sequence.*/

                FileRead readSeqFile = new FileRead();
                fullSeq = readSeqFile.readFullSeq();
                SeqTitle = readSeqFile.readTitle();
                seqLength = readSeqFile.returnLength();

                String EnsemblID = "";                  //Ensembl ID of the query sequence

                /*call readMyID function in the ReadID class
```

```
to get Ensembl ID of the query sequence*/

ReadID readSeqID = new ReadID();
EnsemblID = readSeqID.readMyID();

int seqStart = 0;          //the query sequence start position
int seqEnd = 0;                              //the query sequence end position
int seqDir = 0;                              //the query sequence direction

 /*call functions in SearchID class to get
sequence start position, sequence direction, sequence end position,
Ensembl ID of the query sequence and specie name*/

SearchID searchBlastResult = new SearchID();
seqStart = searchBlastResult.searchRequiredID(EnsemblID);
seqDir = searchBlastResult.returnDirection();
seqEnd = searchBlastResult.searchEnd(seqLength);
EnsemblID = searchBlastResult.returnID();
species = searchBlastResult.returnSpecies();

 int [][] exonLocation = new int[200][2];  //contains all the exon/intron positions

 /*call searchExon function in ExonSearch class to put all exon/intron
 position data into a two dimensional array exonLocation*/

ExonSearch exonPositionSearch = new ExonSearch();
exonPositionSearch.searchExon(exonLocation, EnsemblID, seqStart, seqEnd, species);

int splicingAmount = 0;          //the numbers of splicing sites
int [][] splicingSitePosition = new int[200][2];                  //the position of splicing sites

/*call splicingSite function in SplicingSiteLocation class to get the number of splicing sites
and the location of the splicing sites*/

SplicingSiteLocation siteSplicing = new SplicingSiteLocation();
splicingAmount = siteSplicing.splicingSite(splicingSitePosition, exonLocation, seqLength);
```

```
            int [] A1bindingSeq = new int[1000];  //contains all the location of A1 binding sequence

        /*call searchA1bindingSite function in SearchResult class
          to find out search all A1 binding site sequences.*/

            SearchResult searchMatch = new SearchResult();
            A1bindingSeq = searchMatch.searchA1bindingSite(fullSeq, SeqTitle);

        /*call getResult function in SearchResult to predict A1 binding sites*/
            for (int i=0; i<splicingAmount-1; i++)
            {
                    if (splicingSitePosition[i+1][0]>splicingSitePosition[i][0])
                    {
                            searchMatch.getResult(splicingSitePosition[i][0], splicingSitePosition[i][1],
                                                                    splicingSitePosition[i+1][0], splicingSitePosition[i+1][1]);
                    }
            }

            //remove repeats in the intron array
            int[][] uniqueSite = new int[100][2];
            findUniqueSite uniqueLocation = new findUniqueSite();
            uniqueSite = uniqueLocation.EIsite(splicingSitePosition);

           int [][] A1bindingPredict = new int[50][3]; //contains all the predicted A1 binding locations
            A1bindingPredict = searchMatch.printResult(uniqueSite);  //print the results

        /*Write into tempFile for making figure for A1 binding sites distributions*/
            writeFile fileWrite = new writeFile();
            fileWrite.toFile(seqDir, exonLocation, A1bindingSeq, A1bindingPredict, fullSeq);

    }
}
```

FileRead.java

```
/*read query sequence*/

import java.io.*;

public class FileRead
{

   //read the query sequence and return the query sequence
         public String readFullSeq()
         {
                  try
                  {
                           BufferedReader in = null;
                           String inputLine = "";

                           FileReader reader = new FileReader("FullSequence.txt");
                           in = new BufferedReader(reader);


                           firstLine = in.readLine();     //first line in the query sequence in FASTA form
                           inputLine = in.readLine();     //rest line of the query sequence in FASTA form

                           while (inputLine != null)
                           {
                                    seq = seq + inputLine;
                                    inputLine = in.readLine();
                           }

                           in.close();

                           length = seq.length();      //query sequence length

                  }
                  catch (IOException e)
                  {
```

```
                    System.out.println("Can't open FullSequence.txt file!");
                    System.exit(1);
                }
                return seq;
        }

        //return first line of the query sequence in FASTA form
        public String readTitle()
        {
                return firstLine;
        }

        //return the length of the query sequence

        public int returnLength()
        {
                return length;
        }

        String firstLine = "";
        String seq = "";
        int length = 0;

}
```

ReadID.java

```
/*the Ensembl ID is temporary stored in'tempFile.txt' file by a PHP file called page.php.
  This class reads the Ensembl ID in 'tempFile.txt'*/

import java.io.*;

public class ReadID
{

        public String readMyID()
        {
                String inputLine = "";
                try
                {
                        BufferedReader in = null;


                        FileReader reader = new FileReader("tempFile.txt");
                        in = new BufferedReader(reader);

                        inputLine = in.readLine();

                        in.close();

                 }
                catch (IOException e)
                {
                        System.out.println("Can't open tempFile.txt file!");
                        System.exit(1);
                }
                return inputLine;
        }

}
```

SearchID.java

```
/*Search user selected Ensembl ID of the query sequence in BLAST result
  and find out the start position, end position and strand direction of
  the query sequence.*/

import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.util.StringTokenizer;
import javax.swing.JOptionPane;


public class SearchID
{
        private int startFrom = 0;
        private int endHere = 0;
        String EnID = "";
        int selection = 0;
        int direction = 0;

        public int searchRequiredID(String IDofEnsembl)
        {
                try
                {
                        EnID = IDofEnsembl;

                        BufferedReader in = null;
                        String inputLine = "";
                        String Line1 = "";
                        String Line2 = "";
                        String aFrom = "";
                        String temp = "";
                        String aEnd = "";
```

```
if (EnID.charAt(3) == 'M')
{
        selection = 2;    //Mouse species
        EnID = EnID.substring(0, 18);  //get user selected Ensembl ID
}
else
{
        selection = 1;    //human species
        EnID = EnID.substring(0, 15);
}

FileReader read = new FileReader("c:\\ResearchProj\\Files\\blastResult.txt");
in = new BufferedReader(read);

inputLine = in.readLine();
String inputLineSub = "";

while (inputLine != null)
{
        if (inputLine.length() > 18)
        {
                /*take the first 15 or 18 characters of each line from the BLAST result
                and compare it with user selected Ensembl ID of the query sequence*/

                if (selection == 1)
                        inputLineSub = inputLine.substring(1, 16);
                if (selection == 2)
                        inputLineSub = inputLine.substring(1, 19);

                /*if Ensembl ID is found, combine current line with next line, then igonore
                the next 7 lines (see BLAST result for detail), then read one line into
                Line1 (which contains the first matched character of the query. Igonore
                one line again, and read one line into Line2 which contains the first matched
                character of the hit gene*/

                if ((inputLineSub.equals(EnID))&&((inputLine.charAt(0)) == '>'))
                {
```

```
                        inputLine = inputLine + in.readLine();
                        for (int j=0; j<8; j++)
                        Line1 = in.readLine();
                        Line2 = in.readLine();
                        Line2 = in.readLine();
                        break;
                }
        }
        inputLine = "";
        inputLine = in.readLine();
}

/*Find the absolute number of the start and end position of the query in BLAST result*/
StringTokenizer tokenizer = new StringTokenizer(inputLine);
String token = "";
for (int i=0; i<2; i++)
{
        token = tokenizer.nextToken();
}
aFrom = tokenizer.nextToken();   //first number
token = tokenizer.nextToken();   //ignore next space
temp = tokenizer.nextToken();    //from second number to the next space

for (int i=0; i<temp.length(); i++)
{
        if (temp.charAt(i) != '|')
                aEnd = aEnd + temp.charAt(i);  //second number
        else
                break;
}

if (Integer.parseInt(aFrom) > Integer.parseInt(aEnd))  //this is a reverse sequence
{
        temp = aFrom;
        aFrom = aEnd;
        aEnd = temp;
        direction = 1; //set direction = 1 if it is a reverse sequence
```

```
                }

                StringTokenizer atokenizer = new StringTokenizer(Line1);
                StringTokenizer btokenizer = new StringTokenizer(Line2);
                String atoken = "";
                String btoken = "";

                for (int i=0; i<2; i++)
                {
                        atoken = atokenizer.nextToken();  //position of the first match character in the query
                        btoken = btokenizer.nextToken();  //position of the first match character in the hit gene
                }

                /*correspondent start position of the query sequence in DNA.
                This is a relative number*/
                startFrom = Integer.parseInt(aFrom) + Integer.parseInt(btoken) - Integer.parseInt(atoken);

        }
        catch (IOException e)
        {
                System.out.println("Could not open blastResult.txt file!!!");
                System.exit(1);
        }
        return startFrom;
}

public int searchEnd(int seqLength)
{
        //correspondent end position of the query sequence in DNA.
        endHere = startFrom + seqLength - 1;

        return endHere;
}

public String returnID()
{
        return EnID;
```

```
        }

        public int returnSpecies()
        {
                return selection;
        }
        public int returnDirection()
        {
                return direction;
        }
}
```

ExonSearch.java

```
/*Search ensembl ID in annotated Exon/Intron file ('HumanExon.txt' or 'MouseExon.txt')
  to find out the one that is what the users had selected, then read the each exon start
  and end positions.*/

import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.StringTokenizer;
import java.io.BufferedReader;
import javax.swing.JOptionPane;

public class ExonSearch
{
        public void searchExon(int[][] exonPosition, String seqID, int startHere, int stopHere, int specieSelection)
        {

                try
                {
                        BufferedReader in = null;
                        String Line = "";
                        boolean found = false;
                        String sourceFile = "";

                        if (specieSelection == 1)
                                sourceFile = "c:\\ResearchProj\\SourceFile\\HumanExon.txt";
                        else
                                sourceFile = "c:\\ResearchProj\\SourceFile\\MouseExon.txt";

                        FileReader reader = new FileReader(sourceFile);
                        in = new BufferedReader(reader);

                        Line = in.readLine();
                        Line = in.readLine();
                        String atoken = "";
```

```
String btoken = "";
String ctoken = "";
int start = 0;
int end = 0;
int count = 0;

int firstNumber = 0;
int secondNumber = 0;
int test = 0;


while (Line != null)
{
        if (specieSelection == 1)  atoken = Line.substring(0,15);    //ensembl ID in the annotated exon/intron file
        if (specieSelection == 2)  atoken = Line.substring(0,18);

        int num = 0;
        int n = 0;

        /*Ignore the first 17 or 20 characters of each line which contains
        Ensembl ID and version number*/
        if (specieSelection == 1)  n = 18;
        if (specieSelection == 2)  n = 21;

        for (int i = n; i<Line.length(); i++)
        {
                if (Line.charAt(i) == ',')
                        break;
                else
                        num++;
        }

        //get each exon start and end positions
        if (specieSelection == 1)
        {
                btoken = Line.substring(18, 18+num);    //exon start position
                ctoken = Line.substring(18+num+1, Line.length());   //exon end position
```

```
        }
        if (specieSelection ==2)
        {
                btoken = Line.substring(21, 21+num);
                ctoken = Line.substring(21+num+1, Line.length());
        }

        /*find user selected Ensembl ID and put exon start and end positions
        into exonPosition array*/
        if (seqID.equals(atoken))
        {
                found = true;
                start = Integer.parseInt(btoken);
                end = Integer.parseInt(ctoken);

                if ((start >= startHere) && (start < stopHere))
                {
                        exonPosition[count][0] = start - startHere + 1;
                        //their correspondent exon start position in the sequence

                        exonPosition[count][1] = end - start + exonPosition[count][0];
                        //their correspondent exon end position in the sequence

                        count = count + 1;
                }
        }

        Line = in.readLine();
}//end while


if (found == false)
{
        System.out.println("Could not find your Ensembl ID in the database.");
        System.exit(1);
}
```

```
//sort the content in exonPosition array from small number to large.
int[][] tempArray = new int[1][2];

for (int i=0; i<count; i++)
{
        for (int j=1; j<(count-i); j++)
        {
                if ((exonPosition[j-1][0] > exonPosition[j][0])||
                   ((exonPosition[j-1][0] == exonPosition[j][0])&&(exonPosition[j-1][1] > exonPosition[j][1])))
                {
                        tempArray[0][0] = exonPosition[j-1][0];
                        tempArray[0][1] = exonPosition[j-1][1];

                        exonPosition[j-1][0] = exonPosition[j][0];
                        exonPosition[j-1][1] = exonPosition[j][1];

                        exonPosition[j][0] = tempArray[0][0];
                        exonPosition[j][1] = tempArray[0][1];
                }

        }
}

int p = 0;

//remove repeat contents in the exonPosition array
for (int i = 0; i<count-1;i++)
{
        if ((exonPosition[i][0] == exonPosition[i+1][0]) && (exonPosition[i][1] == exonPosition[i+1][1])&&
           (exonPosition[i][0] != 0))
        {
                p++;

                for (int j=i; j<count-1; j++)
                {
                        exonPosition[j][0] = exonPosition[j+1][0];
```

```
                                        exonPosition[j][1] = exonPosition[j+1][1];

                                    }
                                    exonPosition[count - p][0] = 0;
                                    exonPosition[count - p][1] = 0;
                                }
                            }

                        }
                        catch(IOException e)
                        {
                            System.out.println("Can't open SourceFile for exon search!!!");
                            System.exit(1);
                        }
                        //return reverse;
                    }
}
```

SplicingSiteLocation.java

```
/*Change array containing exon position into array containing intron position.
  splicingArray[0][0] is set to 1, the begining of the sequence,splicingArray[0][1]
  is the end of the 5'-untranslated region.*/

public class SplicingSiteLocation
{

        public int splicingSite(int[][] splicingArray, int[][] exonArray, int lengthSeq)
        {
                int count = 0;
                int num = 0;

                splicingArray[0][0] = 1;
                splicingArray[0][1] = exonArray[0][0];
                num++;

                while (exonArray[count][0] != 0)
                {

                        if (exonArray[count+1][0] > exonArray[count][1])
                        {
                                splicingArray[num][0] = exonArray[count][1];
                                splicingArray[num][1] = exonArray[count+1][0];
                                num = num + 1;
                        }
                        count = count + 1;
                }

                splicingArray[num+1][0] = exonArray[count][1];
                splicingArray[num+1][1] = lengthSeq;
                num++;

                return num;
        }
}
```

SearchResult.java

```
/*Search A1 binding site sequences, SR protein binding site sequences,
  ISSs and ESSs sequences in the query by using Brute-Force algorithm.
  Then make predictions based on the mechanisms by which A1 uses to
  regulate alternative splicing. The parameter setting: overlap = at
  least 1 nt overlap; intron-exon-intron loop structure = 500 nt
  (see thesis for detail)*/

import java.io.*;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import javax.swing.*;

public class SearchResult
{
        String querySeq = "";
        String Title = "";
        int[] locationArray = new int[1000];
        int totalFound = 0;
        int count1 = 0;
        int[][] indication = new int[50][3];
        boolean test = false;
        int sum = 0;

        int[] sf2 = new int[1000];          //put the position of SF2/ASF binding site found
        int[] sc35 = new int[1000];         //put the position of SC35 protein binding site found
        int[] srp20 = new int[1000];        //put the position of SRp20 binding site found
        int[] srp40 = new int[1000];        //put the position of SRp40 binding site found
        int[] srp55 = new int[1000];        //put the position of SRp55 binding site found
        int[] NGE = new int[1000];          //put the position of 9G8 binding site found
        int[] iss = new int[100];          //put the position of ISS  site found
        int[] ess = new int[100];          //put the position of ESS site found

        int scCount = 0;        //how many SC35 binding site found
        int sfCount = 0;        //how many SF2 binding site found
```

```
int issCount = 0;          //how many ISS site found
int essCount = 0;          //how many ESS site found
int srp20Count = 0;        //how many SRp20 binding site found
int srp40Count = 0;        //how many SRp40 binding site found
int srp55Count = 0;        //how many SRp55 binding site found
int NGECount = 0;          //how many 9G8 binding site found

public int[] searchA1bindingSite(String seq, String firstLine)
{
        querySeq = seq;
        Title = firstLine;

        int[] uagggu = new int[200];
        int[] uaggga = new int[200];
        int[] uaggau = new int[200];
        int[] uaggaa = new int[200];
        int[] uagagu = new int[200];
        int[] uagaga = new int[200];
        int[] uagaau = new int[200];
        int[] uagaaa = new int[200];
        int[] uaggca = new int[200];
        int[] uaggcu = new int[200];
        int[] uagauu = new int[200];
        int[] uaguga = new int[200];
        int[] caggga = new int[200];
        int[] uaggua = new int[200];
        int[] uagggc = new int[200];
        int[] uaggag = new int[200];
        int[] gaggga = new int[200];
        int[] uagacu = new int[200];
        int[] uagaag = new int[200];

        char U = 'U';
        char A = 'A';
        char G = 'G';
        char C = 'C';
```

```
        //call searchMatch function to search their correspondent binding sequence
        searchMatch(U, A, G, G, G, U, uagggu);
        searchMatch(U, A, G, G, G, A, uaggga);
        searchMatch(U, A, G, G, A, U, uaggau);
        searchMatch(U, A, G, G, A, A, uaggaa);
        searchMatch(U, A, G, A, G, U, uagagu);
        searchMatch(U, A, G, A, G, A, uagaga);
        searchMatch(U, A, G, A, A, U, uagaau);
        searchMatch(U, A, G, A, A, A, uagaaa);
        searchMatch(U, A, G, G, C, A, uaggca);
        searchMatch(U, A, G, G, C, U, uaggcu);
        searchMatch(U, A, G, A, U, U, uagauu);
        searchMatch(U, A, G, U, G, A, uaguga);
        searchMatch(C, A, G, G, G, A, caggga);
        searchMatch(U, A, G, G, U, A, uaggua);
        searchMatch(U, A, G, G, G, C, uagggc);
        searchMatch(U, A, G, G, A, G, uaggag);
        searchMatch(G, A, G, G, G, A, gaggga);
        searchMatch(U, A, G, A, C, U, uagacu);
        searchMatch(U, A, G, A, A, G, uagaag);

        findSC35();     //call function to find all SC35 binding site in the sequence
        findSF2();      //call function to find all SF2/ASF binding site in the sequence
        findISS();      //call function to find all ISS in the sequence
        findESS();      //call function to find all ESS in the sequence
        findSRp20();    //call function to find all SRp20 binding site in the sequence
        findSRp40();    //call function to find all SRp40 binding site in the sequence
        findSRp55();    //call function to find all SRp55 binding site in the sequence
        find9G8();      //call function to find all 9G8 binding site in the sequence

        return locationArray;
}

public void searchMatch(char letter1, char letter2, char letter3, char letter4, char letter5, char letter6, int[] arr)
{
        int count = 0;
        int temp = 0;
```

```
        for (int i = 0; i < querySeq.length()-6; i++)
        {
                if ((querySeq.charAt(i) == letter1) && (querySeq.charAt(i+1) == letter2) && (querySeq.charAt(i+2) == letter3) &&
                        (querySeq.charAt(i+3) == letter4) && (querySeq.charAt(i+4) == letter5) && (querySeq.charAt(i+5) == letter6))
                {
                        arr[count] = i;
                        count = count + 1;
                }//end if
        }//end for

        for (int i=0; i<count; i++)
        {
            locationArray[totalFound] = arr[i];     //put all A1 binding site in locationArray
            totalFound = totalFound + 1;            //count how many amount it has
        }


        for (int i=0; i<totalFound-1; i++)
        {
                for (int j=0; j<(totalFound - 1 - i); j++)
                {
                        if (locationArray[j] > locationArray[j+1])
                        {
                                temp = locationArray[j];
                                locationArray[j] = locationArray[j+1];
                                locationArray[j+1] = temp;
                        }
                }
        }

}

public void findSC35()
{
         //find SC35 binding site
         for (int i=0; i<querySeq.length()-11; i++)
```

```
{
    //find SC35 (GRYYMCYR  R=A/G, Y=C/U, M=A/C) binding site
    if ((querySeq.charAt(i) == 'G')&&
       (querySeq.charAt(i+1)=='A'||querySeq.charAt(i+1)=='G')&&
       (querySeq.charAt(i+2)=='C'||querySeq.charAt(i+2)=='U')&&
       (querySeq.charAt(i+3)=='C'||querySeq.charAt(i+3)=='U')&&
       (querySeq.charAt(i+4)=='A'||querySeq.charAt(i+4)=='C')&&
       (querySeq.charAt(i+5)=='C'||querySeq.charAt(i+5)=='U')&&
       (querySeq.charAt(i+6)=='C'||querySeq.charAt(i+6)=='U')&&
       (querySeq.charAt(i+7)=='A'||querySeq.charAt(i+7)=='G'))
      {
            sc35[scCount]= i;
            scCount = scCount + 1;
       }

    //find SC35 (UGCYGYY, R=A/G, Y=C/U, M=A/C) binding site
    if ((querySeq.charAt(i)=='U')&&
       (querySeq.charAt(i+1)=='G')&&
       (querySeq.charAt(i+2)=='C')&&
       (querySeq.charAt(i+3)=='C'||querySeq.charAt(i+3)=='U')&&
       (querySeq.charAt(i+4)=='G')&&
       (querySeq.charAt(i+5)=='C'||querySeq.charAt(i+5)=='U')&&
       (querySeq.charAt(i+6)=='C'||querySeq.charAt(i+6)=='U'))
       {
            sc35[scCount]= i;
            scCount = scCount + 1;
      }

    //find SC35 (AGSAGAGUA, S=C/G) binding site
    if ((querySeq.charAt(i)=='A')&&(querySeq.charAt(i+1)=='G')&&
      (querySeq.charAt(i+2)=='C'||querySeq.charAt(i+2)=='G')&&
      (querySeq.charAt(i+3)=='A')&&(querySeq.charAt(i+4)=='G')&&
      (querySeq.charAt(i+5)=='A')&&(querySeq.charAt(i+6)=='G')&&
      (querySeq.charAt(i+7)=='U')&&(querySeq.charAt(i+8)=='A'))
      {
            sc35[scCount]= i;
            scCount = scCount + 1;
```

```
}

//find SC35 (GUUCGAGUA, S=C/G) binding site
if ((querySeq.charAt(i)=='G')&&(querySeq.charAt(i+1)=='U')&&
   (querySeq.charAt(i+2)=='U')&&(querySeq.charAt(i+3)=='C')&&
   (querySeq.charAt(i+4)=='A')&&(querySeq.charAt(i+5)=='G')&&
   (querySeq.charAt(i+6)=='A')&&(querySeq.charAt(i+7)=='G')&&
   (querySeq.charAt(i+8)=='U')&&(querySeq.charAt(i+9)=='A'))
  {
          sc35[scCount]= i;
          scCount = scCount + 1;
  }

//find SC35 (UGUUCSAGWU, S=C/G, W=A/U) binding site
if ((querySeq.charAt(i)=='U')&&(querySeq.charAt(i+1)=='G')&&
   (querySeq.charAt(i+2)=='U')&&(querySeq.charAt(i+3)=='U')&&
   (querySeq.charAt(i+4)=='C')&&(querySeq.charAt(i+5)=='G'||querySeq.charAt(i+5)=='C')&&
   (querySeq.charAt(i+6)=='A')&&(querySeq.charAt(i+7)=='G')&&
   (querySeq.charAt(i+8)=='U'||querySeq.charAt(i+8)=='A')&&
   (querySeq.charAt(i+9)=='U'))
  {
          sc35[scCount]= i;
          scCount = scCount + 1;
  }

//find SC35 (GWUWCCUGCUA, W=A/U) binding site
if ((querySeq.charAt(i)=='G')&&(querySeq.charAt(i+1)=='A'||querySeq.charAt(i+1)=='U')&&
   (querySeq.charAt(i+2)=='U')&&(querySeq.charAt(i+3)=='U'||querySeq.charAt(i+3)=='A')&&
   (querySeq.charAt(i+4)=='C')&&(querySeq.charAt(i+5)=='C')&&
   (querySeq.charAt(i+6)=='U')&&(querySeq.charAt(i+7)=='G')&&
   (querySeq.charAt(i+8)=='C')&&(querySeq.charAt(i+9)=='U')&&
   (querySeq.charAt(i+10)=='A'))
  {
          sc35[scCount]= i;
          scCount = scCount + 1;
  }
```

```
        //find SC35 (GGGUAUGCUG) binding site
        if ((querySeq.charAt(i)=='G')&&(querySeq.charAt(i+1)=='G')&&
           (querySeq.charAt(i+2)=='G')&&(querySeq.charAt(i+3)=='U')&&
           (querySeq.charAt(i+4)=='A')&&(querySeq.charAt(i+5)=='U')&&
           (querySeq.charAt(i+6)=='G')&&(querySeq.charAt(i+7)=='C')&&
           (querySeq.charAt(i+8)=='U')&&(querySeq.charAt(i+9)=='G'))
          {
                  sc35[scCount]= i;
                  scCount = scCount + 1;
          }

        //find SC35 (GAGCAGUAGKS, K=G/U, S=C/G) binding site
        if ((querySeq.charAt(i)=='G')&&(querySeq.charAt(i+1)=='A')&&
           (querySeq.charAt(i+2)=='G')&&(querySeq.charAt(i+3)=='C')&&
           (querySeq.charAt(i+4)=='A')&&(querySeq.charAt(i+5)=='G')&&
           (querySeq.charAt(i+6)=='U')&&(querySeq.charAt(i+7)=='A')&&
           (querySeq.charAt(i+8)=='G')&&(querySeq.charAt(i+9)=='G'||querySeq.charAt(i+9)=='U')&&
           (querySeq.charAt(i+10)=='G'||querySeq.charAt(i+10)=='C'))
          {
                  sc35[scCount]= i;
                  scCount = scCount + 1;
          }

        //find SC35 (AGGAGAU) binding site
        if ((querySeq.charAt(i)=='A')&&(querySeq.charAt(i+1)=='G')&&
           (querySeq.charAt(i+2)=='G')&&(querySeq.charAt(i+3)=='A')&&
           (querySeq.charAt(i+4)=='G')&&(querySeq.charAt(i+5)=='A')&&
           (querySeq.charAt(i+6)=='U'))
          {
                  sc35[scCount]= i;
                  scCount = scCount + 1;
          }
    }

}
```

```
public void findSF2()
{
	//find SF2/ASF binding site
	for (int i=0; i<querySeq.length()-10; i++)
	{
		//find SF2/ASF binding site (CRSMSGW, R=A/G, S=C/G, M=A/C, W=A/U)
		if ((querySeq.charAt(i)=='C')&&
		(querySeq.charAt(i+1)=='G'||querySeq.charAt(i+1)=='A')&&
		(querySeq.charAt(i+2)=='G'||querySeq.charAt(i+2)=='C')&&
		(querySeq.charAt(i+3)=='A'||querySeq.charAt(i+3)=='C')&&
		((querySeq.charAt(i+4)=='C')||(querySeq.charAt(i+4)=='G'))&&
		(querySeq.charAt(i+5)=='G')&&
		((querySeq.charAt(i+6)=='A')||(querySeq.charAt(i+6)=='U')))
		{
			sf2[sfCount] = i;
			sfCount = sfCount + 1;
		}

		//find SF2/ASF binding site (AGGACRRAGC, R=A/G)
		if ((querySeq.charAt(i)=='A')&&
		 (querySeq.charAt(i+1)=='G')&&
		(querySeq.charAt(i+2)=='G')&&
		(querySeq.charAt(i+3)=='A')&&
		((querySeq.charAt(i+4)=='C')||(querySeq.charAt(i+4)=='G'))&&
		((querySeq.charAt(i+5)=='A')||(querySeq.charAt(i+5)=='G'))&&
		((querySeq.charAt(i+6)=='A')||(querySeq.charAt(i+6)=='G'))&&
		(querySeq.charAt(i+7)=='A')&&
		(querySeq.charAt(i+8)=='G')&&
		((querySeq.charAt(i+9)=='C')||(querySeq.charAt(i+9)=='G')))
		{
			sf2[sfCount] = i;
			sfCount = sfCount + 1;
		}

		//find SF2/ASF binding site (RGAAGAAC, R=A/G)
		if ((querySeq.charAt(i)=='A'||querySeq.charAt(i)=='G')&&
		(querySeq.charAt(i+1)=='G')&&(querySeq.charAt(i+2)=='A')&&
```

```
                (querySeq.charAt(i+3)=='A')&&(querySeq.charAt(i+4)=='G')&&
                (querySeq.charAt(i+5)=='A')&&(querySeq.charAt(i+6)=='A')&&
                (querySeq.charAt(i+7)=='C'))
                {
                        sf2[sfCount] = i;
                        sfCount = sfCount + 1;
                }
        }
}

//function for finding all ISS in the sequence
public void findISS()
{
         //find ISS in the sequence
        for (int i=0; i<querySeq.length()-20; i++)
        {
                if ((querySeq.charAt(i)=='G')&&(querySeq.charAt(i+1)=='G')&&
                (querySeq.charAt(i+2)=='C')&&(querySeq.charAt(i+3)=='A')&&
                (querySeq.charAt(i+4)=='G')&&(querySeq.charAt(i+5)=='U')&&
                (querySeq.charAt(i+12)=='G')&&(querySeq.charAt(i+13)=='G')&&
                (querySeq.charAt(i+14)=='C')&&(querySeq.charAt(i+15)=='G')&&
                (querySeq.charAt(i+16)=='A')&&(querySeq.charAt(i+17)=='G')&&
                (querySeq.charAt(i+18)=='G')&&(querySeq.charAt(i+19)=='G'))
                {
                        iss[issCount] = i;
                        issCount = issCount + 1;
                }
        }
}

//function for finding all ESS site in the sequence
public void findESS()
{
        //find ESS (UAGUGAA)  in the sequence
        for (int i=0; i<querySeq.length()-10; i++)
        {
                if ((querySeq.charAt(i)=='U')&&
```

```
                (querySeq.charAt(i+1)=='A')&&
                (querySeq.charAt(i+2)=='G')&&
                (querySeq.charAt(i+3)=='U')&&
                (querySeq.charAt(i+4)=='G')&&
                (querySeq.charAt(i+5)=='A')&&
                (querySeq.charAt(i+6)=='A'))
                {
                        ess[essCount] = i;
                        essCount = essCount + 1;
                }

                //find ESS (CUAGACUAGA) site
                if ((querySeq.charAt(i)=='C')&&(querySeq.charAt(i+1)=='U')&&
                (querySeq.charAt(i+2)=='A')&&(querySeq.charAt(i+3)=='G')&&
                (querySeq.charAt(i+4)=='A')&&(querySeq.charAt(i+5)=='C')&&
                (querySeq.charAt(i+6)=='U')&&(querySeq.charAt(i+7)=='A')&&
                (querySeq.charAt(i+8)=='G')&&(querySeq.charAt(i+9)=='A'))
                {
                        ess[essCount] = i;
                        essCount = essCount + 1;
                }

                //find ESS (UAGGGCAGGC) site
                if ((querySeq.charAt(i)=='U')&&(querySeq.charAt(i+1)=='A')&&
                (querySeq.charAt(i+2)=='G')&&(querySeq.charAt(i+3)=='G')&&
                (querySeq.charAt(i+4)=='G')&&(querySeq.charAt(i+5)=='C')&&
                (querySeq.charAt(i+6)=='A')&&(querySeq.charAt(i+7)=='G')&&
                (querySeq.charAt(i+8)=='G')&&(querySeq.charAt(i+9)=='C'))
                {
                        ess[essCount] = i;
                        essCount = essCount + 1;
                }
        }
}
```

```
//function for finding all SRp20 binding site in the sequence
public void findSRp20()
{
	for (int i=0; i<querySeq.length()-11; i++)
	{
		//find SRp20 binding site (GCUCCUCUUCC) in the sequence
		if ((querySeq.charAt(i)=='G')&&(querySeq.charAt(i+1)=='C')&&
		(querySeq.charAt(i+2)=='U')&&(querySeq.charAt(i+3)=='C')&&
		(querySeq.charAt(i+4)=='C')&&(querySeq.charAt(i+5)=='U')&&
		(querySeq.charAt(i+6)=='C')&&(querySeq.charAt(i+7)=='U')&&
		(querySeq.charAt(i+8)=='U')&&(querySeq.charAt(i+9)=='C')&&
		(querySeq.charAt(i+10)=='C'))
		{
			srp20[srp20Count] = i;
			srp20Count = srp20Count + 1;
		}

		//find SRp20 binding site (CCUCGUCC) in the sequence
		if ((querySeq.charAt(i)=='C')&&(querySeq.charAt(i+1)=='C')&&
		(querySeq.charAt(i+2)=='U')&&(querySeq.charAt(i+3)=='C')&&
		(querySeq.charAt(i+4)=='G')&&(querySeq.charAt(i+5)=='U')&&
		(querySeq.charAt(i+6)=='C')&&(querySeq.charAt(i+7)=='C'))
		{
			srp20[srp20Count] = i;
			srp20Count = srp20Count + 1;
		 }
	}

	for (int i=0; i<querySeq.length()-5; i++)
	{
		//find SRp20 binding site (WCWWC, W=A/U) in the sequence
		if ((querySeq.charAt(i)=='A'||querySeq.charAt(i)=='U')&&
		(querySeq.charAt(i+1)=='C')&&
		(querySeq.charAt(i+2)=='A'||querySeq.charAt(i+2)=='U')&&
		(querySeq.charAt(i+3)=='A'||querySeq.charAt(i+3)=='U')&&
		(querySeq.charAt(i+4)=='C'))
		{
```

```
                    srp20[srp20Count] = i;
                    srp20Count = srp20Count + 1;
                }
        }

        for (int i=0; i<querySeq.length()-7; i++)
        {
                //find SRp20 binding site (CUCKUCY, K=G/U, Y=C/U) in the sequence
                if ((querySeq.charAt(i)=='C')&&(querySeq.charAt(i+1)=='U')&&
                (querySeq.charAt(i+2)=='C')&&
                (querySeq.charAt(i+3)=='G'||querySeq.charAt(i+3)=='U')&&
                (querySeq.charAt(i+4)=='U')&&(querySeq.charAt(i+5)=='C')&&
                (querySeq.charAt(i+6)=='U'||querySeq.charAt(i+6)=='C'))
                {
                        srp20[srp20Count] = i;
                        srp20Count = srp20Count + 1;
                }
        }

}

//function for finding all SRp40 binding site in the sequence
public void findSRp40()
{
        //find SRp40 binding site (UGGGAGCRGUYRGCUCGY, R=A/G, Y=C/U) in the sequence
        for (int i=0; i<querySeq.length()-18; i++)
        {
                if ((querySeq.charAt(i)=='U')&&(querySeq.charAt(i+1)=='G')&&
                (querySeq.charAt(i+2)=='G')&&(querySeq.charAt(i+3)=='G')&&
                (querySeq.charAt(i+4)=='A')&&(querySeq.charAt(i+5)=='G')&&
                (querySeq.charAt(i+6)=='C')&&
                (querySeq.charAt(i+7)=='A'||querySeq.charAt(i+7)=='G')&&
                (querySeq.charAt(i+8)=='G')&&(querySeq.charAt(i+9)=='U')&&
                (querySeq.charAt(i+10)=='U'||querySeq.charAt(i+10)=='C')&&
                (querySeq.charAt(i+11)=='A'||querySeq.charAt(i+11)=='G')&&
                (querySeq.charAt(i+12)=='G')&&(querySeq.charAt(i+13)=='C')&&
                (querySeq.charAt(i+14)=='U')&&(querySeq.charAt(i+15)=='C')&&
```

```
                (querySeq.charAt(i+16)=='G')&&
                (querySeq.charAt(i+17)=='U'||querySeq.charAt(i+17)=='C'))
                {
                        srp40[srp40Count] = i;
                        srp40Count = srp40Count + 1;
                }
        }

        //find SRp40 binding site (YRCRKM, R=A/G, Y=C/U, M=A/C, K=G/U) in the sequence
        for (int i=0; i<querySeq.length()-6; i++)
        {
                if ((querySeq.charAt(i)=='U'||querySeq.charAt(i)=='C')&&
                (querySeq.charAt(i+1)=='G'||querySeq.charAt(i+1)=='A')&&
                (querySeq.charAt(i+2)=='C')&&
                (querySeq.charAt(i+3)=='G'||querySeq.charAt(i+3)=='A')&&
                (querySeq.charAt(i+4)=='G'||querySeq.charAt(i+4)=='U')&&
                (querySeq.charAt(i+5)=='A'||querySeq.charAt(i+5)=='C'))
                {
                        srp40[srp40Count] = i;
                        srp40Count = srp40Count + 1;
                }
        }
}

//function for finding all SRp55 binding site in the sequence
public void findSRp55()
{
        //find SRp55 binding site (YYWCYSG, Y=C/U, W=A/U, S=C/G) in the sequence
        for (int i=0; i<querySeq.length()-7; i++)
        {
                if ((querySeq.charAt(i)=='U'||querySeq.charAt(i)=='C')&&
                (querySeq.charAt(i+1)=='U'||querySeq.charAt(i+1)=='C')&&
                (querySeq.charAt(i+2)=='A'||querySeq.charAt(i+2)=='U')&&
                (querySeq.charAt(i+3)=='C')&&
                (querySeq.charAt(i+4)=='U'||querySeq.charAt(i+4)=='C')&&
                (querySeq.charAt(i+5)=='G'||querySeq.charAt(i+5)=='C')&&
                (querySeq.charAt(i+6)=='G'))
```

```
                {
                        srp55[srp55Count] = i;
                        srp55Count = srp55Count + 1;
                }
        }

}

//function for finding all 9G8 binding site in the sequence
public void find9G8()
{
        for (int i=0; i<querySeq.length()-9; i++)
        {
                //find 9G8 binding site (GACGAC) in the sequence
                if ((querySeq.charAt(i)=='G')&&
                (querySeq.charAt(i+1)=='A')&&
                (querySeq.charAt(i+2)=='C')&&
                (querySeq.charAt(i+3)=='G')&&
                (querySeq.charAt(i+4)=='A')&&
                (querySeq.charAt(i+5)=='C'))
                {
                        NGE[NGECount] = i;
                        NGECount = NGECount + 1;
                }

                //find 9G8 binding site (ACGAGAGAY, Y=C/U)
                if ((querySeq.charAt(i)=='A')&&
                (querySeq.charAt(i+1)=='C')&&
                (querySeq.charAt(i+2)=='G')&&
                (querySeq.charAt(i+3)=='A')&&
                (querySeq.charAt(i+4)=='G')&&
                (querySeq.charAt(i+5)=='A')&&
                (querySeq.charAt(i+6)=='G')&&
                (querySeq.charAt(i+7)=='A')&&
                (querySeq.charAt(i+8)=='C'||querySeq.charAt(i+8)=='U'))
                {
                        NGE[NGECount] = i;
```

```
                    NGECount = NGECount + 1;
                }

                //find 9G8 binding site (WGGACRA, W=A/U, R=A/G)
                if ((querySeq.charAt(i)=='A'||querySeq.charAt(i)=='U')&&
                (querySeq.charAt(i+1)=='G')&&
                (querySeq.charAt(i+2)=='G')&&
                (querySeq.charAt(i+3)=='A')&&
                (querySeq.charAt(i+4)=='C')&&
                (querySeq.charAt(i+5)=='A'||querySeq.charAt(i+5)=='G')&&
                (querySeq.charAt(i+6)=='A'))
                {
                    NGE[NGECount] = i;
                    NGECount = NGECount + 1;
                }
        }
}


//find which A1 binding site sequence is the predicted A1 binding site
 public void getResult(int position1, int position2, int position3, int position4)
 {
        char judge = 'T';      //to avoid repeat number in the final result
        int hexanucleotide = 6;  //A1 binding sequence is a hexanucleotide
        int issLength = 20;    //length of iss
        int overlap = 1;    //at least 1 nt overlap (parameter = 1)

        //Find ESS in the exon.ESS already contains A1 binding sequence.
        for (int i = 0; i<essCount; i++)
        {
            if ((ess[i]>=position2)&&(ess[i]<=position3))
            {
                for (int k=0; k<count1; k++)
                {
                    if (ess[i]==indication[k][0])
                    {
```

```
                                judge = 'F';
                        }
                }
                if (judge == 'T')
                {
                        indication[count1][0] = ess[i];
                        indication[count1][1] = ess[i] + hexanucleotide;
                        indication[count1][2] = 1;          //put a tag 1 to represent binding site is ESS

                        sum = sum + 1;
                        count1 = count1 + 1;
                        test = true;
                }
        }
        judge = 'T';
}


for (int i=0; i<totalFound; i++)
{
        //find overlap between ISS and A1 binding site
        for (int j=0; j<issCount; j++)
        {
                if (((locationArray[i]-iss[j])>=overlap)&&(((iss[j]+issLength)-locationArray[i])>=overlap))
                {
                        for (int k=0; k<count1; k++)
                        {
                                if (indication[k][0]==iss[j])
                                {
                                        judge = 'F';
                                        break;
                                }
                        }
                        if (judge == 'T')
                        {
                                indication[count1][0] = iss[j];
                                indication[count1][1] = iss[j]+issLength;
```

```
                    indication[count1][2] = 2;        //put 2 to represent binding site is in ISS

                    sum = sum + 1;
                    count1 = count1 + 1;
                    test = true;
                }
            }
            judge = 'T';
}

//find A1 binding site located in branch point
int branchpointStart = 20;
int branchpointEnd = 50;
if ((position2>locationArray[i])&&((position2-locationArray[i])<branchpointEnd)
&&((position2-locationArray[i])>branchpointStart))
{
            for (int k=0; k<count1; k++)
            {
                if (locationArray[i]==indication[k][0])
                {
                    judge = 'F';
                    break;
                }
            }
            if (judge == 'T')
            {
                indication[count1][0] = locationArray[i];
                indication[count1][1] = locationArray[i] + hexanucleotide;
                indication[count1][2] = 3;                    //put 3 to indicate that the binding site is close to branch point

                sum = sum + 1;
                count1 = count1 + 1;
                test = true;
            }
}//end if
judge = 'T';
```

```
int edge = hexanucleotide - 1;
//at least 1 nt overlap with the exon (meaning that it is located in exon)

//find overlapp between SC35 and A1 binding site (SC35 or part of SC35 binding site must be in the exon)
int SC35BindingLeng = 11;   //SC35 binding sequence consists of 5 nucleotides
if ((((locationArray[i]-position2)>- edge) && ((position3-locationArray[i]) > 0))
{
        for (int j=0; j<scCount; j++)
        {
                if ((((locationArray[i]-sc35[j])>=overlap)&&((locationArray[i]-sc35[j])<SC35BindingLeng))||
                (((sc35[j]-locationArray[i])>=overlap)&&((sc35[j]-locationArray[i])< hexanucleotide)))
                {
                        for (int k=0; k<count1; k++)
                        {
                                if (locationArray[i]==indication[k][0])
                                {
                                        judge = 'F';
                                        break;
                                }
                        }
                        if (judge == 'T')
                        {
                                indication[count1][0] = locationArray[i];        //first number in A1 binding site
                                indication[count1][1] = locationArray[i] + hexanucleotide;      //last number in A1 binding
                                                                                                //site
                                indication[count1][2] = 4;                      //put 4 to indicate that this is overalap with SC35

                                sum = sum + 1;
                                count1 = count1 + 1;
                                test = true;                                    //indicates that the result is found
                        }
                }
                judge = 'T';
        } //end for
 }//end if

//find overlapped between SF2/ASF and A1 binding site
```

```
int SF2BindingLength = 10;   //SF2 binding sequence contains 10 nucleotides
if (((locationArray[i]-position2)>-edge) && ((position3-locationArray[i]) > 0))
{
        for (int j=0; j<sfCount; j++)
        {
                if ((((locationArray[i]-sf2[j])>=overlap)&&((locationArray[i]-sf2[j])<SF2BindingLength))||
                (((sf2[j]-locationArray[i])>=overlap)&&((sf2[j]-locationArray[i])<hexanucleotide)))
                {
                for (int k=0; k<count1; k++)
                {
                        if (locationArray[i]==indication[k][0])
                        {
                                judge = 'F';
                                break;
                        }
                }
                if (judge == 'T')
                {
                        indication[count1][0] = locationArray[i];
                        indication[count1][1] = locationArray[i] + hexanucleotide;
                        indication[count1][2] = 5;                  //put 5 to indicate that it overlaps with SF2/ASF

                        sum = sum + 1;
                        count1 = count1 + 1;
                        test = true;
                }
        } //end if
        judge = 'T';
}//end for
}//end if

//find overlapped between SRp20 and A1 binding site
int SRp20BindingLeng = 11;    //SRp20 binding sequence contains 11 nucleotides
 if (((locationArray[i]-position2)>-edge) && ((position3-locationArray[i]) > 0))
 {
        for (int j=0; j<srp20Count; j++)
        {
```

```
                    if ((((locationArray[i]-srp20[j])>=overlap)&&((locationArray[i]-srp20[j])<SRp20BindingLeng))||
                    (((srp20[j]-locationArray[i])>=overlap)&&((srp20[j]-locationArray[i])<hexanucleotide)))
                    {
                            for (int k=0; k<count1; k++)
                            {
                                    if (locationArray[i]==indication[k][0])
                                    {
                                            judge = 'F';
                                            break;
                                    }
                            }
                            if (judge == 'T')
                            {
                                    indication[count1][0] = locationArray[i];
                                    indication[count1][1] = locationArray[i] + hexanucleotide;
                                    indication[count1][2] = 6;              //put 6 to indicate that this is SRp20 binding site

                                    sum = sum + 1;
                                    count1 = count1 + 1;
                                    test = true;
                            }
                    }  //end if
                    judge = 'T';
            }//end for
}//end if

//find overlapped between SRp40 and A1 binding site
int SRp40BindingLeng = 10;   //SRp40 binding site contains 10 nucleotides
if (((locationArray[i]-position2)>-edge) && ((position3-locationArray[i]) > 0))
{
            for (int j=0; j<srp40Count; j++)
            {
                    if ((((locationArray[i]-srp40[j])>=overlap)&&((locationArray[i]-srp40[j])<SRp40BindingLeng))||
                    (((srp40[j]-locationArray[i])>=overlap)&&((srp40[j]-locationArray[i])<hexanucleotide)))
                    {
                            for (int k=0; k<count1; k++)
                            {
```

```
                                        if (locationArray[i]==indication[k][0])
                                        {
                                                judge = 'F';
                                                break;
                                        }
                                }
                                if (judge == 'T')
                                {
                                        indication[count1][0] = locationArray[i];
                                        indication[count1][1] = locationArray[i] + hexanucleotide;
                                        indication[count1][2] = 7;                  //put 7 to indicate that this is SPr40 binding site

                                        sum = sum + 1;
                                        count1 = count1 + 1;
                                        test = true;
                                }
                        } //end if
                        judge = 'T';
                }//end for
        }//end if

//find overlapped between SRp55 and A1 binding site
int SRp55BindingLeng = 10;   //SRp55 binding site contains 10 nucleotide
if (((locationArray[i]-position2)>-edge) && ((position3-locationArray[i]) > 0))
{
        for (int j=0; j<srp55Count; j++)
        {
                if ((((locationArray[i]-srp55[j])>=overlap)&&((locationArray[i]-srp55[j])<SRp55BindingLeng))||
                (((srp55[j]-locationArray[i])>=overlap)&&((srp55[j]-locationArray[i])<hexanucleotide)))
                {
                        for (int k=0; k<count1; k++)
                        {
                                if (locationArray[i]==indication[k][0])
                                {
                                        judge = 'F';
                                        break;
                                }
```

```
                        }
                        if (judge == 'T')
                        {
                                indication[count1][0] = locationArray[i];
                                indication[count1][1] = locationArray[i] + 6;
                                indication[count1][2] = 8;              //put 8 to indicate SRp55

                                sum = sum + 1;
                                count1 = count1 + 1;
                                test = true;
                        }
                } //end if
                 judge = 'T';
        }//end for
}//end if

//find overlapped between 9G8 and A1 binding site
int NGEBindingLeng = 10;   //9G8 binding site contains 10 nucleotides
if ((((locationArray[i]-position2)>-edge) && ((position3-locationArray[i]) > 0))
{
        for (int j=0; j<NGECount; j++)
        {
                if ((((locationArray[i]-NGE[j])>=overlap)&&((locationArray[i]-NGE[j])<NGEBindingLeng))||
                (((NGE[j]-locationArray[i])>=overlap)&&((NGE[j]-locationArray[i])<hexanucleotide)))
                {
                        for (int k=0; k<count1; k++)
                        {
                                if (locationArray[i]==indication[k][0])
                                {
                                        judge = 'F';
                                        break;
                                }
                        }
                        if (judge == 'T')
                        {
                                indication[count1][0] = locationArray[i];
                                indication[count1][1] = locationArray[i] + hexanucleotide;
```

```
                                        indication[count1][2] = 9;                  //put 9 to indicate 9G8

                                        sum = sum + 1;
                                        count1 = count1 + 1;
                                        test = true;
                                    }
                                } //end if
                                judge = 'T';
                            }//end for
                    }//end if
}

//find out whether A1 can form multimers to loop out exon
//Here, only 500 nt upstream and downstream of the exon are considered
int beforeNum = 0;      //how many A1 binding site locate upstream of the exon
int afterNum = 0;       //how many A1 binding site locate downstream of the exon
int beforeSpan = 0;     //whether should 500 or the location of the upstream exon be used
int afterSpan = 0;      //whether should 500 or the location of the downstream exon be used

boolean longerIntron1 = false;
boolean longerIntron2 = false;

int checkLength = 500;   //check 500 nt upstream and downstream of interested exon

for (int t=0; t<totalFound; t++)
{
        if ((position2-position1) < checkLength)     //upstream exon locates less than 500 nt
                beforeSpan = position2-position1;
        else
        {
                beforeSpan = checkLength;          //upstream exon locates larger than 500 nt
                longerIntron1 = true;
                        }
        if ((position4-position3) < checkLength)     //downstream exon locates less than 500 nt
                afterSpan = position4-position3;
        else
        {
```

```
                afterSpan = checkLength;            //downstream exon locates larger than 500 nt
                longerIntron2 = true;
        }
        if ((locationArray[t]<position2) && ((position2-locationArray[t])< beforeSpan))
        {
                beforeNum = beforeNum + 1;
        }
        else
        {
                if (((locationArray[t]- position3)< afterSpan) && (locationArray[t] > position3))
                {
                        afterNum = afterNum + 1;
                }
        }
}
if ((beforeNum >= 2) && (afterNum >= 2))    //more than two A1 binding sites in both sides of the exon
{
        for (int k=0; k<count1; k++)
        {
                int repeat = 0;
                if (longerIntron1 == true)
                        repeat = position2 - checkLength;
                else
                        repeat = position2 - position1;

                if (indication[k][0]== repeat)
                {
                        judge = 'F';
                        break;
                }
          }
          if (judge == 'T')
          {
                if (longerIntron1 == true)
                {
                        indication[count1][0] = position2 - checkLength;
                }
```

```
                else
                {
                        indication[count1][0] = position1+1;
                }
                if (longerIntron2 == true)
                {
                        indication[count1][1] = position3 + checkLength;
                }
                else
                {
                        indication[count1][1] = position4-1;
                }

                indication[count1][2] = 10;                //put 10 to indicate that the region can form loop

                sum = sum + 1;
                count1 = count1 + 1;
                test = true;
                longerIntron1 = false;
                longerIntron2 = false;
        }
}
judge = 'T';

//sort the result in indication array from small number to large
int first=0, second=0, third=0;
for (int i=0; i<sum-1; i++)
{
        for (int j=0; j<sum-i; j++)
        {
                if (indication[i][0] > indication[i+1][0])
                {
                        first = indication[i][0];
                        second = indication[i][1];
                        third = indication[i][2];
                        indication[i][0] = indication[i+1][0];
                        indication[i][1] = indication[i+1][1];
```

```
                    indication[i][2] = indication[i+1][2];
                    indication[i+1][0] = first;
                    indication[i+1][1] = second;
                    indication[i+1][2] = third;
                }
            }
        }
}


public int[][] printResult(int[][] uniqueSplicingSite)
{
        try
        {
            PrintWriter out = null;

            FileWriter writer = new FileWriter("c:\\ResearchProj\\Files\\Result.txt");
            out = new PrintWriter(writer);

            out.println();
            out.println("          Report for hnRNP-A1 Binding Site Prediction in pre-mRNA");
            out.println();

            out.println("Input Sequence:");
            out.println();

            out.println(Title);
            out.println();

            out.print("     1  ");

            char c;
            String str = "";

            for (int i=0; i<querySeq.length(); i++)
            {
                if ((i != 0) && ((i%10)==0) && ((i%60) != 0))
```

```
{
        out.print(" " + querySeq.charAt(i));
        c = querySeq.charAt(i);
        str = str + c;
        str = "";
}
else
{
        if ((i != 0) && (((i)%60) == 0))
        {
                out.println("  " + i);

                int digits = 1;
                int division = 0;

                division = i / 10;
                while(division != 0)
                {
                        division = division / 10;
                        digits = digits + 1;
                }

                for (int j = 0; j < (7-digits); j++)
                {
                        out.print(" ");
                }
                out.print((i+1)+"  ");
                out.print(querySeq.charAt(i));
                c = querySeq.charAt(i);
                str = str + c;
                str = "";
        }
        else
        {
                out.print(querySeq.charAt(i));
                c = querySeq.charAt(i);
                str = str + c;
```

```
                        str = "";
                    }
                }
            }
            out.println();
            out.println();
            out.println();

            if (test == false)
            {
                out.println();
                out.println("No possible hnRNP-A1 binding site has been found.");
                out.println();
             }
             else
             {
                out.println("Total of " + sum + " possible hnRNP-A1 binding site(s) are found in your query sequence.");
                out.println("(Exons are expressed in capital letters, while intron in small letters)");
                out.println();

                for (int i=0; i<sum; i++)
                {
                    out.print((i+1) + ")    " + (indication[i][0] + 1) + "  ");

                    if ((indication[i][1] - indication[i][0]) < 20)
                    {
                        for (int j=indication[i][0]; j < indication[i][1]; j++)
                        {
                            out.print(querySeq.charAt(j));
                            c = querySeq.charAt(j);
                            str = str + c;
                            str = "";
                        }
                    }
                    else
                    {
                        for (int k=indication[i][0]; k<indication[i][0]+10; k++)
```

```
		{
			out.print(querySeq.charAt(k));
			c = querySeq.charAt(k);
			str = str + c;
			str = "";
		}
		out.print("  .....   ");

		for (int l=indication[i][1] - 10; l<indication[i][1]; l++)
		{
			out.print(querySeq.charAt(l));
			c = querySeq.charAt(l);
			str = str + c;
			str = "";
		}
}

out.println("  " + (indication[i][1]+1));

String marker = findType(indication[i][2]);
out.println(marker);
out.println();
out.println("SUGGESTION: Please use the following sequence to further confirm the result by checking "
	+ "RNA secondary structure using m-fold (http://www.bioinfo.rpi.edu/applications/mfold/old/rna/).");
out.println();
out.println();

int showStart = 0;
int showEnd = 0;

int distance = indication[i][1] - indication[i][0];

if (distance > 500)
{
		showStart = indication[i][0];
		showEnd = indication[i][1];
}
```

```
else
{
        int n = indication[i][0] - (500 - distance)/2;

        if (n < 0)
                showStart = indication[i][0];
        else
                showStart = n;
        showEnd = showStart + 500;
}
int s = 0;
for (int t=showStart; t<showEnd; t++)
{
        if (((s % 10) == 0) && (s % 60) != 0)
        {
                out.print(" ");
                c = checkSwitch(querySeq, t, uniqueSplicingSite);
                out.print(c);
                str = str + c;
                str = "";
                s = s + 1;
         }
         else
         {
                if (((s % 60) == 0) || (s == 0))
                {
                        if (s != 0)
                        {
                                out.println("  " + t);
                         }

                         int d = t / 10;
                         int p = 0;
                         while (d != 0)
                         {
                                 d = d / 10;
                                 p = p + 1;
```

```
                                        }
                                        for (int x= 0; x<(6-p); x++)
                                        {
                                                out.print(" ");
                                        }
                                        out.print((t+1) + "  ");
                                        c = checkSwitch(querySeq, t, uniqueSplicingSite);
                                        out.print(c);
                                        str = str + c;
                                        str = "";
                                        s = s + 1;
                                }
                                else
                                {
                                        c = checkSwitch(querySeq, t, uniqueSplicingSite);
                                        out.print(c);
                                        str = str + c;
                                        str = "";
                                        s = s + 1;
                                }
                        }
                }
                out.println();
                out.println();
                out.println();
        }//end for
        out.println("THE END OF THE REPORT");
        out.println();
        out.println();
    }
    out.close();
}
catch(IOException e)
{
    System.out.println("Error in opening Result.txt file!");
}
```

```
		return indication;
	}

	public String findType(int n)
	{
		String type = "";

		switch (n)
		{
			case 1:
			{
				type = "This site overlaps with a possible ESS site.";
				break;
			}
			case 2:
			{
				type = "This site overlaps with a possible ISS site.";
				break;
			}
			case 3:
			{
				type = "hnRNP-A1 binding to this site may interfere with protein binding to Branch Point during pre-mRNA
splicing.";
				break;
			}
			case 4:
			{
				type = "This site overlaps with a SC35 binding site.";
				break;
			}
			case 5:
			{
				type = "This site overlaps with a SF2/ASF binding site.";
				break;
			}
			case 6:
			{
```

```
                        type = "This site overlaps with a SRp20 binding site.";
                        break;
                }
                case 7:
                {
                        type = "This site overlaps with a SRp40 binding site.";
                        break;
                }
                case 8:
                {
                        type = "This site overlaps with a SRp55 binding site.";
                        break;
                }
                case 9:
                {
                        type = "This site overlaps with a 9G8 binding site.";
                        break;
                }
                case 10:
                {
                        type = "This site may form a LOOP after hnRNP-A1 binding. Therefore, it contains multiple binding sites for
hnRNP-A1.";
                        break;
                }
        }
        return type;
    }


  //Exons are printed in capital letters, intron in small letters
  //seqSwitch is the query sequence, switchLocation indicates the nucleotide letter to be changed,
  //splicingUnique is an array which doesn't contain repeated splicing site of the sequence.
    public char checkSwitch(String seqSwitch, int switchLocation, int[][] splicingUnique)
    {
        boolean detected = false;

        char character = ' ';
```

```
        int i = 0;

        while (splicingUnique[i][0] != 0)
        {
                if ((splicingUnique[i][0] < switchLocation) && (splicingUnique[i][1] > switchLocation))
                {
                        detected = true;
                        break;
                }
                else
                        i++;
        }

        if (detected == true)
        {
                if (seqSwitch.charAt(switchLocation) == 'A')   character = 'a';
                if (seqSwitch.charAt(switchLocation) == 'G')   character = 'g';
                if (seqSwitch.charAt(switchLocation) == 'C')   character = 'c';
                if (seqSwitch.charAt(switchLocation) == 'U')   character = 'u';
        }
        else
                character = seqSwitch.charAt(switchLocation);

        return character;
    }

}
```

findUniqueSite.java

```
//check array splicingSite which contains all the exon/intron position. Remove repeated exon/intron
//position and put them into exonIntron array.

import java.awt.*;


public class findUniqueSite
{
	public int[][] EIsite(int[][] splicingSite)
	{
		int[][] exonIntron = new int[100][2];
		boolean find = false;
		int n = 0;

		for (int i=0; i<100; i++)
		{
			if (splicingSite[i][0] != 0)
			{
				for (int j=0; j<= n; j++)
				{
					//check splicing site to make them not repeat.
					if ((splicingSite[i][0] == exonIntron[j][0])&&(splicingSite[i][1] == exonIntron[j][1]))
					{
						find = true;
						break;
					}
				}
				if (find == false)
				{
					exonIntron[n][0] = splicingSite[i][0];
					exonIntron[n][1] = splicingSite[i][1];
					n++;
				}
			}
			else
```

```
                                        break;
                        }
                        return exonIntron;
                }
}
```

writeFile.java

```
/*Write exon/intron positions, the location of A1 binding site sequences,
  the locations of the A1 binding sites predicted and the query seqence
  in 'tempFile.txt' for drawing distribution figure by Figure class.*/

import java.io.*;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import javax.swing.*;

public class writeFile
{
        public void toFile(int chainDir, int[][] exonIntronSite, int[] bindingSite, int[][] foundSite, String sequenceQuery)
        {
                try
                {
                        PrintWriter out = null;

                        FileWriter writer = new FileWriter("tempFile.txt");
                        out = new PrintWriter(writer);


                       //write exon/intron positions
                        for (int i = 0; i<200; i++)
                        {
                                if (exonIntronSite[i][0] != 0)
                                {
                                   out.println(exonIntronSite[i][0]);
                                   out.println(exonIntronSite[i][1]);
                                }
                                else
                                {
                                        out.println("###");
                                        break;
                                }
```

```
}

//write predicted A1 binding sites
 for (int i=0; i<1000; i++)
 {
          if (bindingSite[i] != 0)
          {
                   out.println(bindingSite[i]);
          }
          else
          {
                   out.println("$$$");
                   break;
          }
 }

//write the locations of A1 binding site sequences
 for (int i = 0; i<50; i++)
 {
          if (foundSite[i][0] != 0)
          {
                   out.println(foundSite[i][0]);
                   out.println(foundSite[i][1]);
                   out.println(foundSite[i][2]);
          }
          else
          {
                   out.println("%%%");
                   break;
          }
 }

 out.println(chainDir);    //write sequence direction
 out.println(sequenceQuery);   //write query sequence

 out.close();
```

```
                }
                catch(IOException e)
                {
                        System.out.println("Error in opening tempFile.txt file!");
                }
        }
}
```

Figure.java

```
/*Read data from 'tempFile.txt' file
  and draw A1 binding site distribution figure*/

import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.*;

public class Figure extends java.applet.Applet
{
        String FileToRead="tempFile.txt";
        String line;
        String nextLine;
        String chain;
        int n = 0;
        int[][] exonIntron = new int[200][2];
        int[][] tempExonIntron = new int[200][2];
        int[][] bindingSite = new int[1000][2];
        int[][] predictSite = new int[50][3];
        String sequence = "";
        int totalExon = 0;
        int amount = 0;


        public void init()
        {

                // Get setup parameters from applet html
                String param = getParameter("FileToRead");
                if ( param != null)
                {
                        FileToRead = new String(param);
                }
                // Now read the file.
                readFile();
```

```
}

public void readFile()
{
        URL url=null;

        try
        {
                url = new URL (getCodeBase(), FileToRead );
        }
        catch (MalformedURLException  e )
        {
                System.out.println("Malformed URL ");
                stop();
        }

        try
        {
                //read file
                InputStream in=url.openStream();
                BufferedReader dis = new BufferedReader(new InputStreamReader(in));

                line = dis.readLine();

                while (line.compareTo("###") != 0)
                {
                        nextLine = dis.readLine();
                        n = Integer.parseInt(line);

                        exonIntron[totalExon][0] = n;
                        tempExonIntron[totalExon][0] = n;
                        n = Integer.parseInt(nextLine);
                        exonIntron[totalExon][1] = n;
                        tempExonIntron[totalExon][1] = n;
                        totalExon++;

                        line = dis.readLine();
```

```
}

int t = 0;
for (int i=0; i<totalExon; i++)
{
        if ((exonIntron[i][0] == exonIntron[i+1][0])&&exonIntron[i][0] != 0)
        {
                t++;
                exonIntron[i][1] = exonIntron[i+1][1];

                for (int j = i; j<totalExon; j++)
                {
                        exonIntron[j+1][0] = exonIntron[j+2][0];
                        exonIntron[j+1][1] = exonIntron[j+2][1];
                }
        }
}

int p = 0;
for (int i =0; i<totalExon-t; i++)
{
        if ((exonIntron[i][1] == exonIntron[i+1][1])&&(exonIntron[i][1] != 0))
        {
                p++;
                for (int j = i; j<totalExon-t; j++)
                {
                        exonIntron[i+1][0] = exonIntron[i+2][0];
                        exonIntron[i+1][1] = exonIntron[i+2][1];
                }
        }
}

for (int i=0; i<totalExon;i++)
{
        if (exonIntron[i][0] != 0)
                amount++;
        else
```

```
                    break;
}

int count = 0;
line = dis.readLine();
while (line.compareTo("$$$") != 0)
{
        n = Integer.parseInt(line);
        bindingSite[count][0] = n;
        count++;

        line = dis.readLine();
}

count = 0;
String nextNextLine;
line = dis.readLine();
while (line.compareTo("%%%") != 0)
{
        nextLine = dis.readLine();
        nextNextLine = dis.readLine();
        n = Integer.parseInt(line);
        predictSite[count][0] = n;
        n = Integer.parseInt(nextLine);
        predictSite[count][1] = n;
        n = Integer.parseInt(nextNextLine);
        predictSite[count][2] = n;
        count++;

        line = dis.readLine();
}

chain = dis.readLine();
line = dis.readLine();

while (line != null)
{
```

```
                    sequence = sequence + line;
                    line = dis.readLine();
                }

                in.close();
            }
            catch (IOException e )
            {
                System.out.println("Error");
            }

}

//draw distribution figure
public void paint(Graphics g)
{
        Graphics2D g2 = (Graphics2D)g;

        int totalPixel = 550;
        double unit = sequence.length() / totalPixel;
        int currentX = 0;
        int startX = 0;
        int startY = 130;
        int endY = 150;
        int currentLine = 0;
        int endNumber = 0;
        int startNumber = exonIntron[0][0];

        final int size = 24;
        String title = "Exon/Intron position, the location of binding sequence and predicted binding site in the query sequence";
        g2.drawString(title, 50, 20);

        g2.setColor(Color.green);
        int untranslated = 0;

        //draw 5' untranslated region
        if (exonIntron[0][0] > 0)
```

```
{
        if (chain.compareTo("0") == 0)
                untranslated = (int)(exonIntron[0][0]/unit + 0.5);
        else
                untranslated = (int)((sequence.length() - exonIntron[amount-1][1])/unit + 0.5);
        if (untranslated > 0)
                g2.drawLine(startX, (startY+endY)/2, untranslated, (startY+endY)/2);
        else
                untranslated = 0;
}
startX = untranslated;

//draw exon box and intron line
for (int i = 0; i < 200; i++)
{
        if (exonIntron[i][0] > 0)
        {
                if ((exonIntron[i][0] != exonIntron[i+1][0])&&(exonIntron[i][1] != exonIntron[i+1][1]))
                {
                        g2.setColor(Color.red);

                        if (sequence.length() >= exonIntron[i][1])
                                currentX = (int)((exonIntron[i][1] - exonIntron[i][0])/ (unit) + 0.5);
                        else
                                currentX = (int)((sequence.length() - exonIntron[i][0])/unit + 0.5);

                        Rectangle box = new Rectangle(startX, startY, currentX, (endY-startY));
                        g2.draw(box);
                        g2.fill(box);

                        String position = "";
                        position = position + exonIntron[i][0];

                        String site = "";
                        if (sequence.length() > exonIntron[amount-1-i][1])
                                site = site + exonIntron[amount-1-i][1];
                        else
```

```
                        site = site + sequence.length();


                    startX = startX + currentX;

                    position = "";
                    if (sequence.length() > exonIntron[i][1])
                        position = position + exonIntron[i][1];
                    else
                       position = position + sequence.length();

                    g2.setColor(Color.magenta);
                    site = "";
                    site = site + exonIntron[amount-1-i][0];


                    g2.setColor(Color.red);

                    if (exonIntron[i+1][0] > 0)
                    {
                        currentLine = startX + (int)((exonIntron[i+1][0] - exonIntron[i][1]) / (unit) + 0.5);
                        g2.drawLine(startX, (startY+endY)/2, currentLine, (startY+endY)/2);
                        startX = currentLine;
                    }
                }

            }
            else
            {
                endNumber = exonIntron[i-1][1];
                break;
            }
}

//draw 3' untranslated region
int untranslated3 = 0;
```

```
if (sequence.length() > endNumber)
{
        untranslated3 = (int)(sequence.length()/unit+0.5);
        g2.setColor(Color.green);
        g2.drawLine(startX, (startY+endY)/2, untranslated3, (startY+endY)/2);
}

//mark location of hnRNP-A1 binding sequence found in query sequence
g2.setColor(Color.blue);

for (int i = 0; i < 1000; i++)
{
        if (bindingSite[i][0] != 0)
        {
                int location = 0;
                if (chain.compareTo("0") == 0)
                        location = (int)((bindingSite[i][0]) / unit + 0.5);
                else
                        location = (int)((sequence.length() - bindingSite[i][0])/unit +0.5);
                g2.drawLine(location, startY-10, location, startY);
        }
        else
                break;
}

//mark predicted hnRNP-A1 binding sites in the query sequence
g2.setColor(Color.black);
int position = 0;
int n = 1;

for (int i=0; i<1000; i++)
{
        bindingSite[i][1] = 0;
}

for (int i = 0; i< 50; i++)
{
```

```
if (predictSite[i][0] != 0)
{
        for (int j=0; j<1000; j++)
        {
                if (bindingSite[j][0] != 0)
                {
                        if ((bindingSite[j][0] >= predictSite[i][0]) && (bindingSite[j][0] <= predictSite[i][1])
                                && (bindingSite[j][1] == 0))
                        {
                                int place = 0;
                                if (chain.compareTo("0") == 0)
                                        place = (int)((bindingSite[j][0]) / unit + 0.5);
                                else
                                        place = (int)((sequence.length() - bindingSite[j][0]) / unit + 0.5);
                                g2.drawLine(place, endY, place, endY + 10);

                                String t = "";
                                t = t + (bindingSite[j][0]+1);
                                bindingSite[j][1] = 1;

                                if ((((bindingSite[j][0]) / unit) - position) >= 30)
                                {
                                        g2.drawString(t, place, endY + 20);

                                        n = 1;
                                }
                                else
                                {
                                        n++;
                                        g2.drawString(t, place, endY + 20*n);
                                }
                                t = "";
                                position = place;

                        }
                }
                else
```

```
                                        break;
                        }
                }
                else
                        break;
}

g2.setColor(Color.magenta);
String location1 = "";
String location2 = "";
String mark = "";
int x = 300;
int p = 30;
int s = 0;

while (predictSite[s][0] != 0)
{
        location1 = location1 + (predictSite[s][0]+1);
        location2 = location2 + (predictSite[s][1]+1);

        mark = findType(predictSite[s][2]);
        g2.drawString((location1 + " ..... " + location2 + ": " + mark), p, x);

        x = x + 30;
        location1 = "";
        location2 = "";
        s++;
}


final int y = x + 50;
g2.setColor(Color.gray);
g2.drawLine(20, y-30, (int)(1000/unit+0.5)+20, y-30);
g2.drawString("1 kb", (int)(1000/unit+0.5) + 30, y-30);
g2.setColor(Color.green);
g2.drawLine(20, y, 50, y);
g2.drawString("3' or 5' untranslated regions", 60, y);
```

```
        g2.setColor(Color.red);
        Rectangle exonIndicator =  new Rectangle(20, y+20, 20, 30);
        g2.draw(exonIndicator);
        g2.fill(exonIndicator);
        g2.drawString("Exon", 60, y+50);
        g2.drawLine(20, y+90, 50, y+90);
        g2.drawString("Intron", 60, y+90);
        g2.setColor(Color.blue);
        g2.drawLine(20, y+110, 20, y+130);
        g2.drawString("hnRNP-A1 binding sequences found in the query", 60, y+140);
        g2.setColor(Color.black);
        g2.drawLine(20, y+150, 20, y+170);
        g2.drawString("Predicted hnRNP-A1 binding sites in the query", 60, y+170);


}

public String findType(int n)
{
        String type = "";

        switch (n)
        {
                case 1:
                {
                        type = "This site overlaps with a possible ESS site.";
                        break;
                }
                case 2:
                {
                        type = "This site overlaps with a possible ISS site.";
                        break;
                }
                case 3:
                {
                        type = "This site overlaps with a Branch Point.";
                        break;
```

```
        }
        case 4:
        {
                type = "This site overlaps with a SC35 binding site.";
                break;
        }
        case 5:
        {
                type = "This site overlaps with a SF2/ASF binding site.";
                break;
        }
        case 6:
        {
                type = "This site overlaps with a SRp20 binding site.";
                break;
        }
        case 7:
        {
                type = "This site overlaps with a SRp40 binding site.";
                break;
        }
        case 8:
        {
                type = "This site overlaps with a SRp55 binding site.";
                break;
        }
        case 9:
        {
                type = "This site overlaps with a 9G8 binding site.";
                break;
        }
        case 10:
        {
                type = "This site may form LOOP after hnRNP-A1 binding.";
                break;
        }
}
```

```
            return type;
        }
}
```

//This class is for BLAST. It is launched by php in action.php file.

ReadQuerySeq.java

```
import java.io.*;

public class ReadQuerySeq
{
	public static void main(String[] args)
	{
		ReadFile fileReader = new ReadFile();
		fileReader.readfile();

	}
}
```

ReadFile.java

```
/*If user selected a file containing his query sequence, the program will
  call ReadSeqFile class to read the first 1,000 nt and save it in a
  'subSequence.txt' file. If user pastes his query sequence, the first 1,000
  nt of the query sequence is recorded in 'subSequence.txt' file by this class.*/

import java.io.*;


public class ReadFile
{
        public void readfile()
        {
                try
                {
                        BufferedReader in = null;
                        String inputLine = "";
                        String seq = "";
                        String inputSeq = "";
                        String firstLine = "";      //only for the first line of the query sequence in FASTA form

                        FileReader reader = new FileReader("tempFile.txt");
                        in = new BufferedReader(reader);

                        firstLine = in.readLine();

                        String fileName = "1";      //to indicates that user uses a file containing his query

                        if (firstLine.equals(fileName))  //user selected a file containing his query sequence
                        {
                                inputLine = in.readLine();
                                in.close();

                                //call readMe function in ReadSeqFile to record first 1,000 nt of the sequence
                                ReadSeqFile fileRead = new ReadSeqFile();
                                fileRead.readMe(inputLine);
```

```
        }
        else    //user paste his query sequence
        {
                firstLine = in.readLine();

                PrintWriter out = null;

                FileWriter writer = new FileWriter("subSequence.txt");
                out = new PrintWriter(writer);


                out.println(firstLine);

                inputLine = in.readLine();

                while (inputLine != null)
                {
                        seq = seq + inputLine;

                        if (seq.length() <= 1000)
                        {
                                out.println(inputLine);
                        }

                        inputLine = in.readLine();
                }
                out.close();

                in.close();

                /*call fullLengthSeq function in SeqFileMake which change small letter in the
                query sequence into capital letter and DNA sequence into RNA sequence*/
                SeqFileMake wholeSeq = new SeqFileMake();
                wholeSeq.fullLengthSeq(firstLine, seq);
        }
}
catch(IOException e)
```

```
                {
                        System.out.println("Counld not open tempFile.txt file!");
                        System.exit(1);
                }

        }
}
```

SeqFileMake.java

```
/*Read the query sequence. If it is in small letter, change it into capital letter.
  If it is DNA sequence, change it into RNA sequence.*/

import java.io.*;

public class SeqFileMake
{
	public void fullLengthSeq(String title, String content)
	{
		try
		{
			PrintWriter out = null;

			FileWriter writer = new FileWriter("FullSequence.txt");
			out = new PrintWriter(writer);

			String inputSeq = "";

			for (int i=0; i<content.length(); i++)
			{
				if (content.charAt(i) == 'a')
					inputSeq = inputSeq + "A";
				else
				{
					if (content.charAt(i) == 'c')
						inputSeq = inputSeq + "C";
					else
					{
						if (content.charAt(i) == 'g')
							inputSeq = inputSeq + "G";
						else
						{
							if (content.charAt(i) == 't' || content.charAt(i) == 'u'|| content.charAt(i) == 'T')
								inputSeq = inputSeq + "U";
							else
```

```
                                                    inputSeq = inputSeq + content.charAt(i);
                                                }
                                            }
                                        }
                                    }

                                    out.println(title);
                                    out.println(inputSeq);

                                    out.close();
                                }
                                catch (IOException e)
                                {
                                    System.out.println("Could not open FullSequence.txt file!!!");
                                    System.exit(1);
                                }
                        }
                }
```

ReadSeqFile.java

```
/*if a query sequence is longer than 1,000 nt, only first 1,000 nt is written
  into a file called 'subSequence.txt'. Otherwise the full sequence is written
  into the file.*/

import java.io.*;

public class ReadSeqFile
{
        public void readMe(String fileLocation)
        {
                try
                {
                        BufferedReader in = null;
                        String inputLine = "";      //used to read rest of the line of the query sequence
                        String seq = "";                        //is used for counting sequence length which is already read
                        String firstLine = "";      //first line in the query sequence in FASTA form

                        FileReader reader = new FileReader(fileLocation);
                        in = new BufferedReader(reader);

                        PrintWriter out = null;

                        FileWriter writer = new FileWriter("subSequence.txt");
                        out = new PrintWriter(writer);

                        firstLine = in.readLine();    //read the first line of the query sequence
                        out.println(firstLine);

                        inputLine = in.readLine();    //read the rest line of the query sequence

                        while (inputLine != null)
                        {
                                seq = seq + inputLine;

                                /*if a query sequence is longer than 1,000 nt, only the first 1,000 nt
```

```
                    is written into 'subSequence.txt' file*/
                    if (seq.length() <= 1000)
                    {
                        out.println(inputLine);
                    }

                    inputLine = in.readLine();
                }
                out.close();

                in.close();

                /*call fullLengthSeq function in SeqFileMake which change small letter in the
                query sequence into capital letter and DNA sequence into RNA sequence*/
                SeqFileMake wholeSeq = new SeqFileMake();
                wholeSeq.fullLengthSeq(firstLine, seq);

            }
            catch(IOException e)
            {
                System.out.println("Counld not open "+ fileLocation +" file!");
                System.exit(1);
            }
        }
}
```