Modeling and Analysis of Regulatory Elements in *Arabidopsis thaliana* from Annotated Genomes and Gene Expression Data

Amrita Pati

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Computer Science and Applications

Lenwood S. Heath, Ph.D., Chairman

Ruth Grene, Ph.D.

T. M. Murali, Ph.D.

July, 2005
Blacksburg, Virginia

Modeling and Analysis of Regulatory Elements in *Arabidopsis thaliana* from Annotated Genomes and Gene Expression Data

by

Amrita Pati

Committee Chairman: Lenwood S. Heath, Ph.D.

Computer Science

(ABSTRACT)

Modeling of *cis*-elements in the upstream regions of genes is a challenging computational problem. A set of regulatory motifs present in the promoters of a set of genes can be modeled by a biclique. Combinations of *cis*-elements play a vital role in ascertaining that the correct co-action of transcription factors binding to the gene promoter, results in appropriate gene expression in response to various stimuli. Geometrical and spatial constraints in transcription factor binding also impose restrictions on order and separation of *cis*-elements. Not all regulatory elements that coexist are biologically significant. If the set of genes in which a set of regulatory elements co-occur, are tightly correlated with respect to gene expression data over a set of treatments, the regulatory element combination can be biologically directed.

The system developed in this work, XcisClique, consists of a comprehensive infrastructure for annotated genome and gene expression data for *Arabidopsis thaliana*. XcisClique models *cis*-regulatory elements as regular expressions and detects maximal bicliques of genes and motifs, called *itemsets*. An itemset consists of a set of genes (called a geneset) and a set of motifs (called a motifset) such that every motif in the motifset occurs in the promoter of every gene in the geneset. XcisClique differs from existing tools of the same kind in that, it offers a common platform for the integration of sequence and gene expression data. Itemsets identified by XcisClique are not only evaluated for statistical over-representation in sequence data, but are also examined with respect to the expression patterns of the corresponding geneset. Thus, the results produced are biologically directed. XcisClique is also the only tool of its kind for *Arabidopsis thaliana*, and can also be used for other organisms in the presence of appropriate sequence, expression, and regulatory element data. The

web-interface to a subset of functionalities, source code and supplemental material are available online at `http://bioinformatics.cs.vt.edu/xcisclique`.

ACKNOWLEDGMENTS

I express my sincere gratitude to my advisor, Dr. Lenwood S. Heath, for being a constant source of inspiration, support, and guidance. I thank him for patiently listening to all my ideas, meticulously going over cumbersome results, giving valuable direction to my work, and always being there to talk. I am also thankful to my committee members Dr. Ruth Grene and Dr. T. M. Murali. Dr. Grene has provided the much needed biological insight in all stages of this work. Dr. Murali understood my work with the briefest of meetings, had a solution ready for almost any dilemma, and pointed me to the appropriate places to look for more information.

I thank the Department of Computer Science, Virginia Tech, for supporting me through my first year of graduate study. I would like to acknowledge National Science Foundation Grant ITR-0219322. Thanks to the National Center for Biotechnology Information, Nottingham database, and Plant Acting *cis*-Element database for the databases used. Thanks are due to Allan for listening to all my technical problems patiently and helping me work them out. I also thank Cecilia for her help in interpreting the results.

Thanks to Animesh, Kiran, my roommates, everybody in Avataar, and all my friends for being there and making graduate school fun.

Finally, I thank my parents Sanjukta and Aloke Pati for their love and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Biological processes in organisms regulate gene activity through a variety of molecular mechanisms. The process by which information carried by genes is manifested as the phenotype is known as *gene expression*. Gene expression consists of *transcription*, which is the synthesis complementary RNA sequences (mRNA) from DNA, and *translation*, which is the formation of proteins from mRNA. Factors such as biotic and abiotic stress affect the level of gene expression. At any point in time, the intensity of gene expression is measured by the level of mRNA transcripts of that gene. The level of mRNA transcripts is determined by transcription, which is initiated at gene promoters, where, different classes of transcriptional regulators, such as DNA-binding transcription factors, co-activators, co-repressors and chromatin-related proteins, bind to a variety of DNA regulatory elements [Hartl and Jones, 2002]. These DNA regulatory sequences are called *c*is-regulatory elements because they are located on the same strand of DNA as a gene. The binding of transcription factors to *c*is-elements is highly sequence specific and each *c*is-element binds only a subset of transcription factors. Frequently, the binding of more than one transcription factors is required for a gene to be expressed in a certain manner. Therefore, gene-regulation is controlled by modules of *cis*-elements that cause a gene to be expressed in response to stimuli.

Various sources [Kato et al., 2004, Chiang et al., 2003, Pilpel et al., 2001] have indicated the importance of co-occurrence of groups of *cis*-elements. While it is undisputed that the co-occurrence of *cis*-elements is essential for proper binding of transcription factors, it is not yet clear whether the actual order of these elements and the distances between them must be conserved. Pilpel et al.

[2001] have suggested that, while the binding of transcription factors to some *cis*-element modules is markedly affected by their order and separations, the same cannot be said about all *cis*-element modules.

The availability of complete and annotated genomes has increased the significance of sequence analysis. There are several sequence analysis programs that predict promoter regions and several others that predict *cis*-elements and modules of *cis*-elements. However, most of these programs are designed for yeast, mammalian, or vertebrate genomes. The *Arabidopsis thaliana* genome is the first eukaryotic plant genome to be fully sequenced and exhaustively annotated. While a small number of tools predict putative promoters and *cis*-elements in *Arabidopsis thaliana*, none of these identify regulatory element modules that also have functional significance attached to them through analysis of functional genomics data. Programs that use functional genomics data such as chromatin immunoprecipitation (ChIP) data, in other organisms such as yeast, cannot be extended to *Arabidopsis thaliana*, because these data are not available for *Arabidopsis thaliana* on a large scale yet. Gene expression data for *Arabidopsis thaliana* for a wide range of experiments are available in the Nottingham database (`http://www.arabidopsis.info/`). Sequence analysis concurrently with expression data analysis can help identify statistically significant modules of *cis*-elements such that the genes in which they occur exhibit closely correlated expression patterns over certain treatments. This would imply that a particular combination of *cis*-elements is important for gene expression under a given condition or set of conditions.

This thesis identifies a problem of estimating the statistical significance of *cis*-element modules and genesets (defined below), and presents XcisClique, our framework and system for solving the problem. A *motifset* is defined as a set motifs and a *geneset* is defined as a set of genes. An *itemset* is defined as the combination of a geneset and a motifset such that, every motif in the motifset is present in the promoter of every gene in the geneset. Given a set of biologically interesting genes and a set of treatments over which their expression is readable, XcisClique identifies itemsets and for each itemset, calculates correlations between genes of its geneset across the specified set of treatments to assign a measure of tightness of gene expression correlation. Relationships are developed between genesets, motifsets, and sets of treatments. The novelty of the system lies in the feature that evaluates each statistically significant set of genes and motifs with respect to gene expression data, thereby, making it possible to identify biologically relevant results and not results based on sequence analysis alone. Also XcisClique integrates annotated genome data and gene expression data for

Arabidopsis thaliana into a comprehensive system.

The remaining contents of this thesis consist of 8 chapters. Chapter 2 describes the fundamental biological, computational, and statistical concepts employed in this work. It also defines any new terms, concepts, and notations that are used in later chapters. Chapter 3 outlines existing promoter discovery programs and addresses prevalent enumerative methods in putative *cis*-element discovery in detail. In Chapter 4, the actual problem is discussed and formulated, and existing tools that identify regulatory modules are examined with respect to their adaptability to *Arabidopsis thaliana*. Chapter 5 describes data sources and database models for the system developed in this work. Chapters 6 and 7 elaborate the methodologies used in this work. Chapter 8 lists biological case studies involving heat shock genes and cold responsive genes. In the conclusions in Chapter 9, the potential flaws of the system are examined and future directions of research are proposed. Appendix A lists all the *cis*-elements curated for use in XcisClique. Appendix B lists all Affymetrix slides from which expression data has been extracted and used. Appendices C and D contain Perl, MATLAB and Shell scripts that comprise XcisClique's programs.

Chapter 2

Preliminaries

This chapter provides the background needed in the remainder of the thesis. Section 2.1 defines the biological entities involved in transcriptional regulation and the computational and statistical concepts used to model the same, as well as new terms and notations. Section 2.2 describes the biological details of gene transcription.

2.1 Definitions

This section defines biological (Section 2.1.1), computational (Section 2.1.2), and statistical (Section 2.1.3) terms.

2.1.1 Biological Terms

DNA

DNA or *deoxyribonucleic acid* is a molecule that encodes genetic information and is found within large structures called **chromosomes**. DNA consists of linear chains of nucleotides, each nucleotide being one of the bases, adenine (A), thymine (T), guanine (G) or cytosine (C). The sequence of these bases encodes genetic information, which is used for gene expression [Brooker, 1999]; by transcription factors to recognize and bind to *cis*-elements during transcription, and to synthesize poly-peptide sequences during translation.

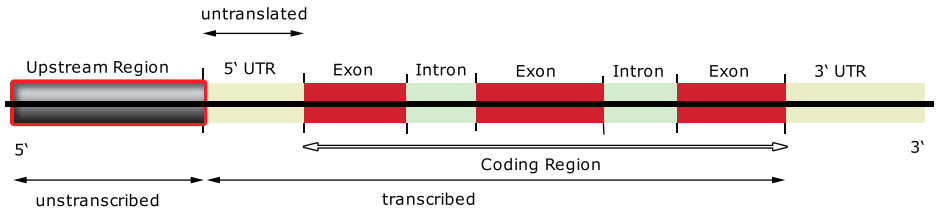


Figure 2.1: Organization of the eukaryotic gene

Gene

A *gene* is a unit of heredity that may influence the outcome of an organism's traits [Brooker, 1999]. Wilhelm Johannsen, a Danish botanist, coined this term in 1909. A gene is located in a specific site of a chromosome as a segment of DNA and hence, consists of a sequence of nucleotides. Genes encode information necessary for the synthesis of proteins that are responsible for many biological processes and hence phenotypic traits of an organism. Figure 2.1 illustrates the structure of a eukaryotic gene. The eukaryotic gene starts with an upstream region. This is followed by a non-coding region called the 5' untranslated region (UTR). Following this is the coding region that consists of alternating sequences of exons and introns. During transcription, both exons and introns are transcribed into RNA, but the introns are excised from the RNA sequence by a process called splicing. The coding region is also called the *open reading frame (ORF)*. The 3' untranslated regions marks the 3' end of the gene. The UTRs are transcribed but not translated.

In this thesis, we indicate *Arabidopsis thaliana* genes by Arabidopsis Gene Identifier (AGI) numbers of the form $AtYgXXXXX$ where $1 \leq Y \leq 5$ and indicates the chromosome that the gene is located on, and $0 \leq X \leq 9$.

Promoter

The region of the gene upstream of the coding and 5' UTR regions is called the *promoter* and is also known as the *upstream region* [Weaver and Hedrick, 1997] or the *regulatory region* of the gene. Figure 2.4 illustrates the structure of a eukaryotic promoter. The nucleotide at the transcription start site (TSS) has a position 0. Nucleotides upstream of the TSS have positions on the negative scale depending on how many bases upstream of the TSS they are. For example, a nucleotide 30

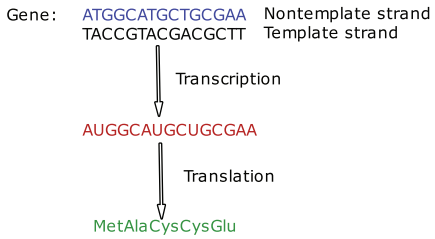


Figure 2.2: Transcription of DNA into mRNA

bases upstream of the TSS has a position of −30. The details will be explained in Section 2.2.

Transcription

Transcription is the first step in the synthesis of protein from DNA and is the process by which RNA (called messenger RNA or mRNA) that is complementary to one of the DNA strands is made. Figure 2.2 illustrates this.

The bottom strand that serves as a template for transcription is called the template strand or transcribed strand. The mRNA carries the same genetic code as the top strand. The top strand is also called the 'sense' or 'coding' strand and the bottom strand is called the 'anti-sense' or 'non-coding' strand. Initiation, regulation and termination of transcription are controlled by RNA polymerase and proteins called *transcription factors*. These will be described in detail in Section 2.2.

Transcription Factors

Transcription Factors (TFs) are proteins or protein complexes that bind to specific DNA sequences called *cis*-elements located in the gene promoter. Transcription factors are either general or gene-specific [Weaver and Hedrick, 1997]. General transcription factors help initiate transcription by delivering RNA polymerases to their respective gene promoters and maintain the basal level of transcription. Gene specific transcription factors either increase the rate of transcription by binding to specific DNA sequences called enhancers or inhibit transcription by binding to specific DNA sequences called silencers (ref. Section 2.1.1).

Regulon

The set of all genes regulated by a transcription factor is known as the *regulon* of that transcription factor [Fowler et al., 2005].

cis-elements

The term *cis* is a Latin word meaning 'here'. *'cis'* and *'trans'* are used to describe the two ways genetic loci interact [Weaver and Hedrick, 1997]. *cis*-acting elements, also known as *cis*-elements, *regulatory elements (RE)* or *transcription factor binding sites (TFBS)*, are specific DNA sequences that are located in the promoter of a gene and serve as binding sites of transcription factors (ref. Section 2.1.1). *cis*-elements are so called because they have to be on the same strand of the DNA molecule as a gene in order to influence the rate of transcription of that gene. *cis*-elements that enhance the rate of transcription when transcription factors bind to them are known as *enhancers*, and those that reduce the rate of transcription are called *silencers*. An example of a *cis*-element in *Arabidopsis thaliana* is the Heat Shock Element (HSE) to which heat shock transcription factors bind in order to elicit response under heat stress. Table 2.1 outlines a few other *cis*-elements in *Arabidopsis thaliana*.

Homologous, Orthologous and Paralogous Genes

Two genes or proteins are said to be *homologous* if they evolved from a common ancestor gene or protein. *Orthologous genes* are homologous genes (or *orthologs*) in different species. *Paralogous genes*, on the other hand, are homologous genes (or *paralogs*) in the same organism created through gene duplication events.

Gene Expression

Gene expression is the process by which information contained in genes is decoded to produce other molecules that determine the phenotypic traits of organisms [Hartl and Jones, 2002]. Gene expression in response to a stimulus at a particular time point is measured by the level of mRNA transcribed. The most widely used tools for measuring gene expression include one-channel and two-channel microarrays, and SAGE (Serial Analysis of Gene Expression) NCBI Expression Resources.

Treatment

A *treatment* is defined as an environmental condition, growth medium, pathogen attack or any other stimulus applied to an organism to elicit a response.

2.1.2 Computational Terms and Concepts

Alphabet

An alphabet is a finite, non-empty set of symbols [Hopcroft et al., 2001]. An alphabet is represented by the symbol Σ. Examples of alphabets are:

1. $\Sigma = \{0, 1\}$, the binary alphabet.
2. $\Sigma = \{a, b, \ldots, z\}$, the Roman alphabet.

Each nucleotide in a DNA sequence can be represented by the following letters: A, C, G, T, U, R, Y, S, W, K, M, B, D, H, V, N, which are the IUPAC symbols for nucleotides [Brooker, 1999]. Table 2.2 explains their respective meanings. The alphabet used to denote DNA sequences will be

$$\Sigma_{DNA} = \{A, C, G, T, U, R, Y, S, W, K, M, B, D, H, V, N\}$$

and will be used throughout this document.

String

Hopcroft et al. [2001] have defined a *string* as a finite sequence of symbols chosen from Σ. The *length of a string* is defined as the number of positions for symbols in that string [Hopcroft et al., 2001]. 00110010 is a string of length 8 over the binary alphabet. The empty string is denoted by ϵ and is the unique string of length 0. The set of all strings over an alphabet is denoted by Σ^*. The set of all strings of length w over an alphabet is represented by Σ^w.

A DNA sequence is a continuous chain of elements from Σ_{DNA}. A DNA sequence of finite length can be represented by a string over Σ_{DNA}. The length of the DNA sequence represented by the string $ACGTTTGCA$ is 9. In all places in this document, a *DNA string* will mean a string that represents a DNA sequence.

Complementarity

A DNA molecule is structured as a double helix where, two DNA strands are twisted around a common axis, in reverse orientation. This double-stranded structure is stabilized by hydrogen bonding between base pairs. An adenine (A) base in one strand always bonds to a thymine (T) base in another strand, and a guanine (G) base on one strand always bonds to a cytosine (C) base on another strand. This is known as Chargaff's rule [Brooker, 1999] and implies that, base sequences in the two DNA strands are *complementary* to one other.

We can define a symmetric function

$$f^c : \Sigma_{DNA} \mapsto \Sigma_{DNA}$$

such that, $f^c(b)$ = the complement of b $\forall b \in \Sigma_{DNA}$. Therefore, the set of ordered pairs in f^c is $\{(A,T),(T,A),(C,G),(G,C),(A,U),(U,A),(R,Y),(Y,R),(S,S),(W,W),(K,M),(M,K),(B,V),(V,B),(D,H),(H,D),(N,N)\}$.

The *complement* of a string S over Σ_{DNA} is defined as the string S^c obtained by complementing the element at every position of S. If S is a string of length n,

$$S^c(i) = f^c(S(i)),\ 1 \leq i \leq n. \tag{2.1}$$

where $S(i)$ indicates the element at position i in string S.

The *reverse complement* of a string S over Σ_{DNA} is defined as the string S^{RC} obtained by reversing S and complementing the element at every position of the reversed string. If S is a string of length n,

$$S^{RC}(i) = f^c(S(n-i+1)),\ 1 \leq i \leq n. \tag{2.2}$$

For example, the reverse complement of the string $ATGCGGC$ is the string $GCCGCAT$.

Palindrome

A string S over Σ_{DNA} is called a *palindrome* if

$$S = S^{RC}$$

Note that only strings of even length can be palindromes. An example of a palindrome is the string $ATGCAT$.

Motif

A *motif* is a string of nucleotides or amino-acids that is hypothesized to have a biological significance. In the context of this thesis, a *motif* or *regulatory motif* will connote a string that represents the DNA sequence of a *cis*-element.

Regular Expression

A *regular expression* is an algebraic description of a programming language in which some important applications such as text search applications or compiler components are expressed (adapted from Hopcroft et al. [2001]). The regular expressions over an alphabet $\mathbf{\Sigma}^*$ are all strings over the alphabet $\mathbf{\Sigma} \cup \{(.), \varnothing, \cup, *\}$ that can be obtained as follows [Lewis and Papadimitriou, 2002].

1. $\varnothing$ and each member of Σ is a regular expression.
2. If α and β are regular expressions, then so is $(\alpha\beta)$.
3. If α and β are regular expressions, then so is $(\alpha \cup \beta)$.
4. If α is a regular expressions, then so is α^*.
5. Nothing is a regular expression unless it follows from (1) through (4).

Every regular expression represents a language, according to the interpretation of the symbols $\cup$ and $*$ as set union and Kleene star, and of juxtaposition of expressions as concatenation. The relation between regular expressions and the languages they represent is established by a function $\mathcal{L}.$ such that if α is any regular expression, then $\mathcal{L}(\alpha)$ is the language represented by α. So, $\mathcal{L}$ is a function from strings to languages. The function $\mathcal{L}$ is defined as follows.

1. $\mathcal{L} = \emptyset$, and $\mathcal{L}(a) = \{a\}$ for each $a \in \Sigma$.
2. If α and β are regular expressions, then $\mathcal{L}((\alpha\beta)) = \mathcal{L}(\alpha)\mathcal{L}(\beta)$.
3. If α and β are regular expressions, then $\mathcal{L}((\alpha \cup \beta)) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$.
4. If α is a regular expression, then $\mathcal{L}(\alpha^*) = \mathcal{L}(\alpha)^*$.

Regular expressions are closely related to non-deterministic finite automata and can model the same language as various forms of automata. However, regular expressions also provide a declarative

way to express the strings we want to accept and are therefore, valuable for describing searches for interesting patterns. An example of a regular expression in Perl is $AGC(G|T)$ that matches strings $AGCG$ and $AGCT$.

Consensus Sequences, Patterns, and Matches

A *cis*-element is a regulatory sequence where transcription factors bind preferentially. Consider a **heat shock element (HSE)** that binds a **heat shock transcription factor (HSF)**. In response to heat stress, HSF of higher eukaryotes is converted from a monomeric form to a trimeric form capable of high-affinity binding to HSE and transcriptional activation [Schoffl et al., 1998]. A biochemically stable binding of the HSF to the HSE consists of protein-protein interactions including hydrophobic interactions and extensive hydrogen bonding. An HSE consists of alternating repeats of the trinucleotide GAA, separated by 2 nucleotides. The defining sequence for the HSE motif is therefore, $GAANNTTCNNGAA$ or its reverse complement, $TTCNNGAANNTTC$, which can also be represented by the regular expressions $GAA[ACTG]\{2\}TTC[ACTG]\{2\}GAA$ and $TTC[ACTG]\{2\}GAA[ACTG]\{2\}TTC$, respectively. However, binding of an HSF to an HSE can tolerate mutations in the A and T residues in the above motifs [Santoro et al., 1998]. If exactly 1 mutation is allowed in the HSE, an HSE can be represented by any of the regular expression below:

$$GA[CGT][ACGT]\{2\}TTC[ACTG]\{2\}GAA$$
$$G[CGT]A[ACGT]\{2\}TTC[ACTG]\{2\}GAA$$
$$GAA[ACGT]\{2\}[ACG]TC[ACTG]\{2\}GAA$$
$$GAA[ACGT]\{2\}T[ACG]C[ACTG]\{2\}GAA$$
$$GAA[ACGT]\{2\}TTC[ACTG]\{2\}G[CGT]A$$
$$GAA[ACGT]\{2\}TTC[ACTG]\{2\}GA[CGT]$$
$$(A|C|G)TC[ACGT]\{2\}GAA[ACGT]\{2\}TTC$$
$$T[ACG]C[ACGT]\{2\}GAA[ACGT]\{2\}TTC$$
$$TTC[ACGT]\{2\}G[CGT]A[ACGT]\{2\}TTC$$
$$TTC[ACGT]\{2\}GA[CGT][ACGT]\{2\}TTC$$
$$TTC[ACGT]\{2\}GAA[ACGT]\{2\}[ACG]TC$$
$$TTC[ACGT]\{2\}GAA[ACGT]\{2\}T[ACG]C$$

Hence an HSE with exactly 1 mutation can be represented by the regular expression
$(GA[CGT][ACGT]\{2\}TTC[ACTG]\{2\}GAA)|(G[CGT]A[ACGT]\{2\}TTC[ACTG]\{2\}GAA)|(GAA$

$[ACGT]\{2\}[ACG]TC[ACTG]\{2\}GAA)|(GAA[ACGT]\{2\}T[ACG]C[ACTG]\{2\}GAA)|(GAA[ACG$
$T]\{2\}TTC[ACTG]\{2\}G[CGT]A)|(GAA[ACGT]\{2\}TTC[ACTG]\{2\}GA[CGT])|((A|C|G)TC[ACG$
$T]\{2\}GAA[ACGT]\{2\}TTC)|(T[ACG]C[ACGT]\{2\}GAA[ACGT]\{2\}TTC)|(TTC[ACGT]\{2\}G[CG$
$T]A[ACGT]\{2\}TTC)|(TTC[ACGT]\{2\}GA[CGT][ACGT]\{2\}TTC)|(TTC[ACGT]\{2\}GAA[ACGT$
$]\{2\}[ACG]TC)|(TTC[ACGT]\{2\}GAA[ACGT]\{2\}T[ACG]C)$.

A *cis*-element can be defined as a regular expression that represents a set of strings. This regular expression is also known as a *consensus sequence*. Any element of the set of strings represented by the regular expression is a match and the length of that element is known as the length of the match.

Bipartite Graph

A *bipartite graph* $G = (U, V, E)$ is a graph whose vertex set can be partitioned into non-empty sets U and V such that the edge set $E = \{(u, v)|u \in U \& v \in V\}$.

Complete Graph

A *complete graph* or *clique* $G(V, E)$ is defined as a graph in which each pair of vertices is connect by an edge. A complete graph with n vertices has $\frac{n(n-1)}{2}$ edges.

Biclique

A *biclique* is a complete bipartite graph. It can be defined as an undirected graph that can be divided into two partitions, i.e., the vertices of the graph can be divided into two non-empty sets A and B such that every edge in the graph goes between A and B, and every such edge is included. Figure 2.3 depicts a biclique with edges $(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5)$.

Itemset, Geneset, and Motifset

Consider a bipartite graph $G(U, V, E)$ such that every vertex $u \in U$ represents a motif, every vertex $v \in V$ represents a gene, and every edge $e = (u, v) \in E$ represents the presence of motif u in the promoter of gene v. The biclique thus formed, is known as an *itemset*. The set of vertices representing genes in the itemset is known as the *geneset* and the set of vertices representing motifs in the itemset is known as the *motifset*. Every edge in an itemset goes between the geneset and the motifset and every such edge is included, implying that in an itemset, every motif in the motifset

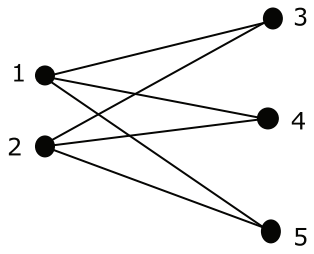


Figure 2.3: Example of a biclique

is present in the promoter of every gene in the geneset. Itemsets are described in more detail in Section 7.4.

Set of Upstream Regions

The upstream region of a gene G is represented by the notation $\mathcal{U}(G)$. The set of upstream regions of a set of genes $\mathcal{G}$ is represented by the notation $\mathcal{U}(\mathcal{G})$.

Presence of a Motif in an Upstream Region

The upstream region $\mathcal{U}(G)$ of a gene G can be represented by a finite DNA string over the alphabet Σ_{DNA}. A motif can also be represented by a string M over the alphabet Σ_{DNA}. M is said to be present in $\mathcal{U}(G)$ if $M \sqsubset \mathcal{U}(G)$, where $\sqsubset$ denotes the "substring" operator. A set of motifs $\mathcal{M}$ is said to be present in the upstream region of a gene G if a match can be found in $\mathcal{U}(G)$ for the strings representing each of the motifs in $\mathcal{M}$. This statement can be extended to define the presence of a motifset in the upstream regions of a geneset.

We define the operation "present in", denoted by $\exists X_m(\mathcal{U}(X_g))$ that represents the presence of X_m in X_g, where X_m is a motif or a motifset and X_g is a gene or a geneset. The presence of a motifset $\mathcal{M}$ in the upstream regions $\mathcal{U}(\mathcal{G})$ of a set of genes $\mathcal{G}$ is represented by the following notation:

$$\exists \mathcal{M}(\mathcal{U}(\mathcal{G}))$$

Consider the set $\mathcal{G}$ of all genes from the entire *Arabidopsis thaliana* genome and a set of motifs $\mathcal{M}$ such that $\exists \mathcal{M}(\mathcal{U}(G)) \; \forall \; G \; \in \; \mathcal{G}$. $\mathcal{G}$ is therefore the set of all genes in the *Arabidopsis thaliana*

genome whose upstream regions can be characterized by the presence of motifset $\mathcal{M}$. We indicate such a set of genes for a motifset $\mathcal{M}$ by $\mathcal{C}_{\mathcal{M}}$.

Foreground and Background Sequences

Given an analysis A in XcisClique, the inputs to the system are a set of genes G, and a set of *cis*-elements M. In this context, the set of upstream regions $\mathcal{U}(G)$, of the genes in G is the foreground set, and is denoted by $\mathcal{F}(A)$. The background set is the set of upstream regions in the entire genome, and is indicated by $\mathcal{B}(A)$. The concept of foreground and background sets is defined for genes as well as upstream sequences.

In the context of an analysis A, the choice of $\mathcal{B}(A)$ might be done in different ways. For example, consider an input set of genes G, whose members are genes regulated by transcription factor T_G. $\mathcal{B}(A)$ can be the set of genes whose regulation is not affected in any way by T_G.

2.1.3 Statistical Terms and Concepts

Population

A *population* is a set of objects that have some characteristic in common [Sheskin, 2000]. For example, all the genes in *Arabidopsis thaliana* comprise a population. The notation N will represent the total number of subjects or objects in the population.

Sample

A *sample* is a set of objects that have been selected from a population [Sheskin, 2000]. In research, a random sample is ideal for drawing inferences about the population from which it was sampled.

Parameter

A *parameter* refers to a characteristic of a population [Sheskin, 2000]. Mean and variance are examples of parameters.

Statistic

A *statistic* refers to a characteristic of a sample [Sheskin, 2000]. Examples of statistics for a random sample of N genes drawn from the *Arabidopsis thaliana* genome are:

1. Average length of each gene.

2. Sum of fold changes of all genes in response to a treatment.

The statistical methods employed in this work use data from one or more samples to draw inferences or make predictions with respect to the population on the basis of a statistic.

Mean

The *mean* is the most common measure of central tendency and is defined as the average score in a distribution. The mean of a random variable X in a sample of size n is represented as:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{2.3}$$

where X_i represents the i^{th} occurrence of the random variable X. In this work, we will frequently refer to the mean of a random variable X as $\mu(X)$ and μ_X.

Standard Deviation

The *standard deviation* is the most common measure of variability. The standard deviation of a population is denoted by σ and is calculated as follows:

$$\sigma = \frac{\sum_{i=1}^{n} (X - \mu)^2}{N} = \frac{\sum_{i=1}^{n} X^2 - \frac{(\sum_{i=1}^{n} X)^2}{N}}{N} \tag{2.4}$$

When the standard deviation of a sample is calculated for inferential purposes (which is done in this work), it is denoted by $\tilde{s}$ and is an unbiased estimate of the population standard deviation. It is calculated as follows:

$$\tilde{s} = \frac{\sum_{i=1}^{n} (X - \bar{X})^2}{n-1} = \frac{\sum_{i=1}^{n} X^2 - \frac{(\sum_{i=1}^{n} X)^2}{n}}{n-1} \tag{2.5}$$

The standard deviation of a sample calculated for descriptive purposes is denoted by $\bar{s}$ and calculated as follows:

$$\tilde{s} = \frac{\sum_{i=1}^{n} (X - \bar{X})^2}{n} = \frac{\sum_{i=1}^{n} X^2 - \frac{(\sum_{i=1}^{n} X)^2}{n}}{n} \tag{2.6}$$

We will also indicate the standard deviation of a random variable X by $\sigma(X)$ and σ_X.

Sample Space

The *sample space* is the set Ω of all possible outcomes of a random experiment. Each experimental result is a single point in the sample space.

Event

Let ω be a sample space.

1. A *d-system* is a family of subsets containing Ω and closed under proper difference (if $A, B \in \mathcal{D}$ and $A \subseteq B$, then $B \setminus A \in \mathcal{D}$) and countable increasing union.

2. A *π-system* is a family of subsets closed under finite intersection.

A class that is both a π-system and a d-system is a *σ-algebra*. An *event* is a subset of Ω. The family of events associated with a random experiment with sample space Ω is a σ-algebra $\mathcal{F}$ on Ω [Karr, 1993]. An *elementary event* or *simple event* consists of a single outcome in the sample space while a *compound event* consists of more than one outcome.

Probability

Probability is a countably additive *set function* defined for events. The *countably additive* property implies that the probability of the countable disjoint union of events is the sum of their individual probabilities [Karr, 1993]. A σ-algebra on Ω is a family of subsets of Ω containing Ω and closed under countable union, countable intersection, and complementation. Let $(\Omega, \mathcal{F})$ be a sample space and a σ-algebra of events.

A *probability* on $(\Omega, \mathcal{F})$ is a function $P : \mathcal{F} \to \mathbb{R}$ such that

1. $P(A) \geq 0$ for all $A \in \mathcal{F}$.

2. $P(\Omega) = 1$.

3. Whenever $A_1, A_2, \ldots$ are (pairwise) disjoint sets in $\mathcal{F}$,

$$P\big(\sum_{n=1}^{\infty} A_n\big) = \sum_{n=1}^{\infty} P\big(A_n\big). \tag{2.7}$$

The frequentist definition computes the probability of an event A by taking the ratio of the number of outcomes favorable to A, n_A, to the number of possible outcomes, n:

$$P(e) = \frac{n_A}{n}. \tag{2.8}$$

An empirical approach to probability is the relative frequency approach. If an experiment is conducted n times and an event A occurs n_A number of times, then the probability of A is approximately:

$$P(e) \approx \frac{n_A}{n}. \tag{2.9}$$

The *probability density function (PDF)* is defined as the point mass at $\omega \in \Omega$ and is denoted by $f(A)$. The *cumulative distribution function (CDF)* is the function $F_A : \mathbb{R} \to [0, 1]$ defined by

$$F_A(t) = P((-\infty, t]).$$

Independent Events

We define the *indicator function* of the set $A \subseteq \Omega$ as the function on Ω given by

$$\mathbf{1}_A(\omega) = \begin{cases} 1 & \text{if } \omega \in A \\ 0 & \text{if } \omega \notin A. \end{cases} \tag{2.10}$$

Thus, $\mathbf{1}_A$ "indicates" whether A occurs [Karr, 1993]. Set operations on events are generalized by algebraic operations on indicator functions:

$$\mathbf{1}_{A \cup B} = \max\{\mathbf{1}_A, \mathbf{1}_B\} \tag{2.11}$$

$$\mathbf{1}_{A \cap B} = \min\{\mathbf{1}_A, \mathbf{1}_B\} \tag{2.12}$$

$$\mathbf{1}_{A_c} = 1 - \mathbf{1}_A, \ A_c \text{ is the complement of } A \tag{2.13}$$

$$\mathbf{1}_{A \Delta B} = |\mathbf{1}_A - \mathbf{1}_B|, \ \Delta \text{ is the symmetric difference operator.} \tag{2.14}$$

Events $A_1, A_2, \ldots, A_n$ are *independent* if their indicator functions are independent random variables [Karr, 1993]. Events $A_1, A_2, \ldots, A_n$ are independent if and only if

$$P(\bigcap_{i \in I} A_i) = \prod_{i \in I} P(A_i) \tag{2.15}$$

for every subset I of $\{1, \ldots, n\}$.

Spearman Correlation Coefficient

The Spearman rank correlation coefficient is a measure of correlation between ordinal variables. It is indicated by ρ and explained in detail in Section 6.1.2. The Spearman rank correlation coefficient between genes g_1 and g_2 will be indicated by $\rho(g_1, g_2)$.

Hypotheses, Null Hypotheses, Alternate Hypotheses

A research *hypothesis* is a general statement of what a researcher predicts [Sheskin, 2000]. An example of a hypothesis is: "Genes in *Arabidopsis thaliana* that respond to heat stress, have a different distribution of heat shock elements (HSEs) in their promoters, as compared to the rest of the genes in the genome". To evaluate this hypothesis, it is restated within a framework of two hypotheses: null and alternate. The *null hypothesis* is a statement of no effect or difference and is represented by the notation H_0. Let μ_1 be the average number of HSEs per gene in the genes that respond to heat stress and μ_2 be the average number of HSEs per gene in all other genes of the *Arabidopsis thaliana* genome. The null hypothesis for the above hypothesis is: "Genes in *Arabidopsis thaliana* that respond to heat stress have the same distribution of heat shock elements (HSEs) in their promoters, as the rest of the genes in the genome" or $H_0 : \mu_1 = \mu_2$. The *alternate hypothesis* indicates the presence of a difference and is represented by H_1. The alternate hypothesis in the above case is: "Genes in *Arabidopsis thaliana* that respond to heat stress have a different distribution of heat shock elements (HSEs) in their promoters, as compared to the rest of the genes in the genome" or $H_1 : \mu_1 \neq \mu_2$.

Test Statistic, Sampling Distribution, and Statistical Significance

Hypothesis testing in inferential statistics yields a *test statistic* (adapted from Sheskin [2000]). A test statistic is evaluated with reference to a *sampling distribution*, which is a theoretical probability distribution of all possible values the test statistic can assume if one were to conduct an infinite number of studies employing a sample size equal to that used in the experiment being evaluated. The test statistic is interpreted by using tables that contain extreme or critical values of the statistic. These values are highly unlikely to occur if the null hypothesis is true. These values allow us to determine whether a result is statistically significant.

The term *statistical significance* implies determination of whether an obtained difference in results in an experiment is due to chance or is the result of a genuine experimental effect.

Hypergeometric Distribution p-value

Consider a population comprised of N of which C objects possess the specific property ϕ we are interested in. Suppose we select a sample of n objects from this population. Let X be the random variable representing the number of objects with property ϕ in a sample. Then the probability of

finding exactly c objects with property ϕ, in this sample, is given by the hypergeometric distribution as:

$$P_r[X = c] = \mathcal{H}(N, C, n, c) = \frac{\binom{n}{c} * \binom{N-n}{C-c}}{\binom{N}{C}} \tag{2.16}$$

The hypergeometric distribution is discussed in greater detail in Section 6.2.

Sum of Absolute Values (SAV) Statistic

The sum of absolute values of all values of a random variable X in a sample of size n is a statistic represented by S and computed as follows:

$$S = \sum_{i=1}^{n} |X_i|. \tag{2.17}$$

Statistics for Correlation with All Genes

We will use the following notations to indicate the mean and standard deviations, respectively of the correlation coefficients ρ of a gene G with the rest of the genes of the *Arabidopsis thaliana* genome. The mean of the Spearman correlation coefficients of a gene G with the remaining genes of the genome is $\mu_{(G,N,\rho)}$ and the standard deviation of the Spearman correlation coefficients of a gene G with the remaining genes of the genome is

$$\sigma_{(G,N,\rho)}. \tag{2.18}$$

***z*-score**

Consider a random variable X that is normally distributed. The distance of X from the mean ($\mu(X)$), in terms of standard deviation ($\sigma(X)$) units, is represented by the random variable Z with mean 0 and standard deviation 1. This is called the Z-score.

$$Z = \frac{X - \mu(X)}{\sigma(X)}. \tag{2.19}$$

The probability of X having a value of x ($P_r[X = x]$) is equal to the probability of Z taking up a value of $z = \frac{x-\mu(X)}{\sigma(X)}$ and is defined as the *standard normal probability density function* denoted by $\mathfrak{n}(X)$.

The probability of X having a value less than or equal to x ($P_r[X <= x]$) is equal to the probability of Z taking up a value of $z <= \frac{x-\mu(X)}{\sigma(X)}$ ($P_r[Z <= z]$) and is defined as the *standard*

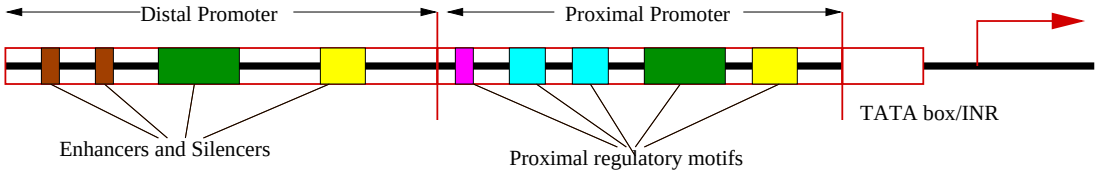


Figure 2.4: Organization of the eukaryotic promoter, adapted from Werner [1999].

normal cumulative distribution function denoted by $\mathfrak{N}(X)$. The probability of X having a value greater than x ($P_r[X > x]$) is equal to the probability of Z taking up a value of $z > \frac{x-\mu(X)}{\sigma(X)}$ ($P_r[Z > z]$) and is defined as the *tail probability* of the normal distribution of X, and is equal to $1 - \mathfrak{N}(X)$.

2.2 Transcriptional Regulation

Gene transcription is complex in eukaryotes. Eukaryotic cells are large and compartmentalized into organelles and there are numerous genes encoding cellular proteins. Multicellular higher eukaryotes require that genes be transcribed in the right kind of cell during the proper stage of development. Also, transcriptional regulation in eukaryotes follows a combinatorial logic, resulting in a vast diversity of gene regulatory activities. The eukaryotic promoter structure is responsible to a large extent for the regulation of transcription. The typical eukaryotic promoter consists of the structural elements described as in Figure 2.4.

2.2.1 Basal Promoter

The basal or core promoter is responsible for guiding the polymerase to the correct transcription start site (TSS) [Weaver and Hedrick, 1997]. Positions upstream of the TSS are counted downwards starting at -1 and positions downstream of the TSS are counted upwards starting at +1, the TSS being the +1 position. Transcription can be initiated accurately only if the pre-initiation complex (PIC) is assembled correctly. This complex consists of the basal transcription factors, which include RNA Polymerases, GTFs (general transcription factors), the TBP (TATA-binding protein), TBP-associated factors (TAFs), and other factors that make up the basal transcription complex. The TATA box and the initiator (INR) are the main sequences through which the core promoter interacts with transcription factors. The TATA box is an AT-rich region near position -30 and is the most

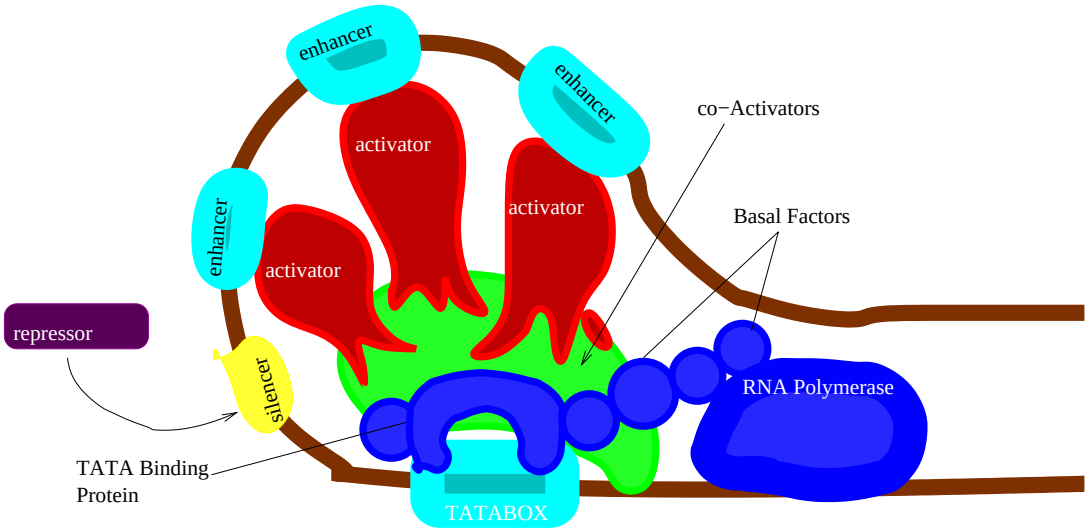


Figure 2.5: Transcription in eukaryotes happens by the cooperative binding of a number of transcription factors to DNA. This picture was inspired by the ideas in Andel et al. [1999] and [Tjian, 1995].

prominent sequence in eukaryotic promoters. When the TBP binds to the TATA box, the DNA is bent such that the sequences upstream and downstream of the TATA box are brought closer. Following this, TFIIB, the TFIIF-polII complex, TFIIE, and TFIIH are recruited by protein-protein interactions which orients the PIC towards the TSS. Figure 2.5 shows the proteins involved in transcription in eukaryotes. TFIIH triggers the transcription by modifying a pol-II subunit and unwinding the DNA double helix in a 10 base pair long stretch downstream of the TSS. However, not all promoters contain a TATA box. In such promoters, the initiator sequence (which is present instead of the TATA box) mediates the binding of TFIID.

2.2.2 Proximal Promoter

The proximal promoter is located upstream of the core promoter. The *cis*-elements to which general TFs (Section 2.1.1) bind are located in this region. These regulatory motifs can promote or suppress the binding of the core promoter components involved in the basal transcription complex. Sequence-specific DNA-binding TFs are capable of activating or repressing transcription. They cause genes to be selectively expressed in response to a certain environmental condition in a particular cell. These

factors have been grouped into families based on their DNA-binding functional domains. In contrast to the basal TFs, co-regulators, and other TFs have diverged to a large extent. Some TFBSs such as TATA boxes and C/EBP binding sites are non-specific and are found in most genes. Examples are the CCAAT box and the GC box binding proteins that serve to increase the production rate of the basal machinery irrespective of the gene in which they are present. Other TFs are specific in their activity and typically bind to promoters of genes that are regulated in response to some stimulus such as a pathogen or an environmental condition. Promoters of a set of co-regulated genes tend to share the same *c*is-elements and their arrangement [Pilpel et al., 2001, Wasserman and Sandelin, 2004, Nardone et al., 2004, Hudson and Quail, 2003]. These approaches have been explained in Sections 3.2.1 and 4.4. TF binding sites in eukaryotes are typically between 5 and 15 bp long [Riechmann, 2002, Ohler, 2001, Fickett and Hatzigeorgiou, 1997]. Geometric constraints dictate that these binding sites be spaced at particular distances from each other based on the spatial requirements of the TFs binding to them. There is also a dependence between the DNA sequence at the binding site of one transcription factor and the ability of that factor to interact with another TF. The precise mechanism by which promoter sequence differences are translated into different receptivity to specific transcription factors is not yet known but sequence specificity and conformational changes are likely to be involved.

2.2.3 Distal Promoter

The distal promoter contains *cis*-elements known as either enhancers or repressors. It is located up to thousands of base pairs away from the TSS, downstream of the gene, and also in the introns of the gene [Weaver and Hedrick, 1997]. Conceivably, the TFs that bind to these elements also bind to the basal transcription complex by bending the DNA into a loop and thus, affecting transcription. These sequences can suppress or enhance transcription significantly and the strand on which they occur does not affect their influence [Weaver and Hedrick, 1997]. They are also responsible for tissue-specific expression as are elements in the proximal promoter, often consist of repeats of the same elements as in the proximal promoter. An enhancer can affect the transcription of more than one gene, both by changing the chromatin structure and by interacting with the PIC. Example of an enhancer in *Arabidopsis thaliana* is the AGCBOXNPGLB *cis*-element which is the binding site for Arabidopsis ethylene response factors that function as activators of GCC box-dependent transcription [Lelievre et al., 1992].

Table 2.1: *Cis*-regulatory elements in *Arabidopsis thaliana*

***cis*-element**	**Consensus Sequence**	**Annotation**
MYBCORE	CNGTTR	Binding site for at least two plant MYB proteins ATMYB1 and ATMYB2, both isolated from Arabidopsis. Involved in dehydration, water stress, flavonoid biosynthesis in root and shoot.
IBOX	GATAAG	Conserved sequence upstream of light-regulated genes, binding site of LeMYB1 which is a member of a novel class of myb-like proteins.
ABREATRD22	RYACGTGGYR	ABA Responsive Element in Arabidopsis dehydration responsive gene rd22.
JASE1ATOPR1	CATACGTCGTCAA	JASE2 found in the promoter of AT 12-oxo-phytodienoic acid-10,11-reductase (OPR1) gene. Involved in up-regulation by both senescence and Jasmonic Acid.
ARFAT	TGTCTC	Auxin Response Factor binding site found in the promoters of primary/early auxin response genes of AT.

Table 2.2: IUPAC codes for nucleic acids

Character	**Meaning**
A	Adenine
C	Cytosine
G	Guanine
T	Thymine
U	Uracil
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base

Chapter 3

Promoter Analysis

In the last decade, a number of computational tools have been designed and implemented for analysis of promoters in various organisms. These tools broadly fall into two categories. Tools in the first category are used to discover promoter regions of genes from nucleotide sequences. These will be described further in Section 3.1. Tools in the second category discover putative *cis*-elements in the promoters of a family of genes using pattern discovery and pattern matching techniques and are described in Section 3.2.

3.1 Promoter Discovery

In this section, we outline some frequently used methods for detecting promoter regions. Transcription initiation is caused by the binding of TFs to TFBSs in the proximal promoter (Ref. 2.2.2). Fickett and Hatzigeorgiou [1997] give an overview of existing tools and techniques for promoter prediction in eukaryotes and state that "the primary computational approach to promoter recognition has been to combine modules recognizing individual binding sites, using some overall description of how these sites should be partially arranged". Many available programs use PWMs (Position Weight Matrices) to model binding sites. PWMs assign a weight to each possible nucleotide at each position of a putative binding site. Building PWMs requires sufficient information about the binding of a TF to various mutated forms of a TFBS. The principal public source of PWMs is the TRANSFAC database [Wingender et al., 1997] that has PWMs corresponding to only 10 *Arabidopsis thaliana* binding sites. PWMs are also generated by motif discovery programs such as AlignACE, MEME and

MotifSampler. However, the use of PWMs is restricted by their availability. Examples of existing tools that use this approach are Autogene (`ftp.bionet.nsc.ru`) and PromoterScan [Prestridge, 1995]. NNPP [Reese and Eeckman, 1995] combines recognition of the TATA box and the initiator region, using a time-delay neural network architecture that allows for variable spacing between the features. A pruned neural network gives clues about the importance of specific positions in the promoter element.

Another approach to recognizing promoter regions exploits differences in nucleotide hexamer frequencies of promoters, protein coding regions, and introns. Promfind [Hutchinson, 1996], which is designed for vertebrate promoter sequences, takes training sets from GenBank and the Eukaryotic Promoter Database (EPD, described below). A discriminant measure separates the promoter regions from the coding and non-coding regions. The Dragon Promoter Finder program [Bajic et al., 2002] predicts TSS positions in DNA sequences of vertebrates using sensors for functional regions (promoters, introns and exons) and an artificial neural network. Experimentally determined eukaryotic POL II promoters with transcription start sites (TSSs) are available in the Eukaryotic Promoter Database (EPD) [Praz et al., 2002]. EPD contains only 14 *Arabidopsis thaliana* promoters whose TSSs have been determined.

Werner [1999] describes models for eukaryotic promoters, and categorizes promoter prediction tools into *general promoter recognition* and *specific promoter recognition* tools. Tools for the former consist of programs that infer promoter location based solely on core promoter properties as well as programs that infer promoter location based on both the core and proximal promoter properties. Typically, such programs produce promoter models based on training sets derived from the Eukaryotic Promoter Database [Praz et al., 2002] and various sets of non-promoter sequences. These promoter models are general and do not require any particular information about a given promoter to make a prediction. Specific promoter recognition tools are based on the occurrence of modules of two or more elements in proximal promoters. These programs are helpful only when promoters of a specific class are of interest. Older promoter prediction programs did not perform well because they took into account only one or two features, such as the presence of the TATA box or the initiator element, but did not consider characteristics conferred by sequence and structure. Recent programs take many more features into consideration, are sequence and structure-based, and therefore, have better accuracy. Rombauts et al. [2003] describe these more recent promoter prediction tools. Recently Shahmuradov et al. [2005] have developed a new program TSSP-TCM for the prediction

of plant promoters that also provides the confidence of the prediction. Using their program, they annotate promoters in the whole Arabidopsis genome and find that the predicted promoters are in good agreement with the start position of known Arabidopsis mRNAs. The program predicted promoters for 18601 genes (out of 27128 annotated ones) in the genome. Results are located at `http://mendel.cs.rhul.ac.uk/` and might be used for further verification and experimental studies of the regulation of plant genes. Almost all the other programs referenced above are restricted to mammalian, vertebrate, or *Drosophila* sequences. This tool only predicts promoter regions and does not predict regulatory motifs.

3.2 Motif Discovery

In the discovery of putative regulatory motifs from promoter sequence data, two principal approaches are used by existing computational models: *enumerative* and *probabilistic*. Section 3.2.1 reviews enumerative models while Section 3.2.2 reviews probabilistic methods. XcisClique uses an enumerative approach for identifying itemsets, hence this chapter describes the existing enumerative methods in more detail than existing probabilistic methods.

3.2.1 Enumerative Approaches

Enumerative approaches work by counting small sequence motifs and identifying motifs that are statistically significant. Motifs are identified either by pattern matching or by constructing suffix trees. Below, we elucidate existing programs.

Regulatory Sequence Analysis Tools (RSAT)

van Helden et al. [1998] extract regulatory motifs from yeast upstream regions using computational and statistical analysis of oligonucleotide frequencies. Let A be an analysis in RSAT and G be a set of genes of interest to be analyzed, such that $|G| = n$. Then, $\mathcal{F}(A) = \mathcal{U}(G) = \{u_1, u_2, \ldots, u_n\}$. $\mathcal{B}(A)$ is the set of all promoters of length 800 bases in the entire yeast genome. Let $L_B = 800 * \mathcal{B}(A)$ denote the total length of the background sequences. The expected frequencies of all possible oligonucleotides of length between 1 and 9 bases is estimated from the background set.

$$f_x = \text{fraction of length } L_B \text{ in } \mathcal{B}(A) \text{ that are } x,\ x \in \Sigma^w_{DNA},\ 1 \leq w \leq 9. \tag{3.1}$$

where x denotes an oligonucleotide. The random variable X represents the expected number of occurrences of an oligonucleotide x in $\mathcal{U}(G)$.

$$E[X] \;=\; 2f_x \sum_{i=1}^{n} (L_i - w + 1) \tag{3.2}$$

where L_i is the length of u_i and w is the length of x. If $L_i \;=\; L \;\forall\; i$, we can define the total number of possible matching positions for an oligonucleotide of length w across both strands as

$$T \;=\; 2n(L - w + 1) \tag{3.3}$$

and hence,

$$E[X] \;=\; f_x \;*\; T. \tag{3.4}$$

Assuming X follows a binomial distribution with parameters T and f_x, the probability of X being greater than or equal to k is

$$P_r[X \geq k] \;=\; \sum_{i=k}^{T} \binom{T}{i} (f_x)^i (1 - f_x)^{T-i} \tag{3.5}$$

Let R be an equivalence relation on Σ_{DNA}^{w} such that

$$R \;=\; \{(x, y) \mid x = y \text{ or } x = y^c\}$$

Palindromes fall under this equivalence class. Then, the number of distinct oligonucleotides of length w when a search is performed on both strands, is

$$D_w \;=\; \left|\Sigma_{DNA}^{w} \backslash R\right| \;=\; \frac{4^w - (4^w - N_{pal})}{2} \tag{3.6}$$

where $N_{pal} = 4^{w/2}$ is the number of possible palindromes. van Helden et al. [1998] state that "a good criterion to conserve an oligonucleotide of length w is to conserve only those oligonucleotides whose occurrence probability is lower than $1/D_w$". The significance coefficient of an oligonucleotide of length w is

$$X_{sig} = -log_{10}[P_r[X \geq k] \times D_w] \tag{3.7}$$

The user chooses the cutoff for X_{sig} for motif discovery. van Helden et al. [1998] consider X_{sig} values of 1 or greater, as significant. Larger positive values of sig, imply a greater degree of over-representation of the motif. This process is exhaustive and rigorous, but it recognizes only short motifs with a highly conserved core.

In *Arabidopsis thaliana*, only some motifs have a short conserved core. A considerable number of motifs are longer and have conserved repeats rather than conserved cores. For instance, the heat shock element consists of 3 or more repeats of conserved trinucleotides (GAA or TTC) separated by $2-5$ nucleotides. This means a minimum motif length of 13 bases and a maximum length of 19 bases if we consider just 3 repeats. Another such motif is CARGATCONSENSUS (sequence $CCWWWWWWGG$). XcisClique does not address motif discovery in *Arabidopsis thaliana*. Discovery of novel motifs that are represented by regular expressions is still an open computational problem.

Automatic Detection of Regulatory Motifs

Jensen and Knudsen [2000] calculated statistics for motifs by three different paradigms as follows.

Hypergeometric statistics on sequence patterns

Consider an analysis A in this model. Let G be the set of all 6269 annotated yeast genes. $\mathcal{U}(G)$ is defined as the set of 200 bases long sequences starting at position -300 of each of the genes in G. The model assumes that regulatory motifs are also present in downstream sequences. If $\mathcal{D}(G)$ is the set of all 200 bases long sequences starting 100 bases downstream of the ORFs of genes in G, then

$$\mathcal{F}(A) \cup \mathcal{B}(A) \;=\; \mathcal{U}(G) \cup \mathcal{D}(G) \tag{3.8}$$

The inputs to the algorithm are a positive set of genes G^+, a negative set of genes G^-, and their corresponding sequences $S^+ = \mathcal{U}(G^+) \cup \mathcal{D}(G^+)$ and $S^- = \mathcal{U}(G^-) \cup \mathcal{D}(G^-)$, respectively. Let M be the set of all motifs of length 1 to 200, that are present in S^+. An algorithm identifies motifs that are overrepresented in S^+ as compared to S^-. Suffix trees are used to count matches of motifs in M in S^+ and S^-. Both strands are considered separately. This identifies motifs that have a preference for one orientation.

To determine if a motif $m \in M$ of length w is significant, the following counts are obtained.

1. P_t =Number of sequences in S^+ with m
2. N_f =Number of sequences in S^+ without m
3. P_f =Number of sequences in S^- with m
4. N_t =Number of sequences in S^- without m

The positive set is a sample of size $(P_t + N_f)$ from the pool of all sequences $(P_t + P_f + N_t + N_f)$ containing $(P_t + P_f)$ sequences with the motif. P_t is assumed to have a hypergeometric distribution. The exact significance potential of m being overrepresented is:

$$m_\alpha = -log(1 - \mathcal{H}(P_t + P_f + N_t + N_f, P_t + P_f, P_t + N_f, P_t)^{(4^w)}) \tag{3.9}$$

All motifs with $m_\alpha > 4$ are analyzed.

Systematic analysis of functional annotation

The input to this model is a set G of 5097 yeast genes and their functional annotations. $\mathcal{U}(G)$ is defined as the set of 500 bases long sequences upstream of the ORF of all genes in G. Jensen and Knudsen [2000] define a *pseudo-sequence* as a continuous string of alpha-numeric characters only, devoid of white spaces. The functional annotation of each gene $g \in G$ is converted into a pseudo-sequence $\int(g)$, and a set D of all k-tuples of length up to 10 that occur in at least two pseudo-sequences (about 10000) is made. For each $d \in D$, G^+ is defined as a set of genes such that $d \sqsubset \int(g) \; \forall \; g \in G^+$, and G^- is defined as the set of genes such that d not $\sqsubset \int(g) \; \forall \; g \in G^-$. S^+ and S^- are constructed as in the previous method and motif-wise significance potentials are calculated for all motifs in M as in Equation 3.9.

Correlating motifs to DNA array data

In their work, Jensen and Knudsen [2000] also correlate the presence of a motif to DNA expression at a specific time point. For the time point being analyzed, the ratio of treated to control for each gene is calculated and their upstream regions are assigned a rank based on this ratio. They use the Kolmogorov-Smirnov ratio [Jensen and Knudsen, 2000] to make the correlation.

The methods used by Jensen and Knudsen [2000] have some apparent shortcomings. Firstly, in Equation 3.9, the purpose of raising the hypergeometric probability to the number of words of length w possible is not justified and the assumption made in doing so is not stated clearly. Second, correlating the presence of a motif to expression data from a single time point is not very useful. One of the reasons is that expression data for a single time point does not capture the behavior of the gene over a period of time. It is not possible to relate the presence of a motif to gene expression over a set of treatments.

YMF

Sinha and Tompa [2002] propose another enumerative method for discovering novel TFBSs. This model is also tailored to accurately represent TFBS in yeast, and the genesets used in their analysis are well-studied yeast regulons. They define a motif as a string of length 6-10 bps with 0 or more consecutive N residues inserted at the center and a limited number of R, Y, S and W characters. This tool is known as Yeast Motif Finder (YMF).

Let Y be the set of all genes in the yeast genome. Let $\mathcal{M}$ be the the 3^{rd} order Markov Chain whose transitions are determined by $4-$mer frequencies in $\mathcal{U}(Y)$. $\mathcal{M}$ is precomputed and used for generating the background distribution for motif discovery.

Let G be a set of n genes input to YMF, such that $\mathcal{U}(g) = l, \forall\ g \in G$. Then, the set of foreground sequences S^+ is equal to $\mathcal{U}(G)$ and the set of background sequences S^- is $\{s \mid s \text{ is randomly generated by } \mathcal{M},\ \text{length}(s) = l,\ \text{and } |S^-| = n\}$.

Consider a motif m whose occurrence in G is being analyzed. Let X_m be the random variable that denotes the number of occurrences of m in S^-. Let $\mu(X_m)$ and $\sigma(X_m)$ denote the mean and standard deviation of X_m respectively. Let N_m be the number of occurrences of m in S^-. Then, the $z-$score associated with m is

$$\mathcal{Z}_m = \frac{N_m - \mu(X_m)}{\sigma(X_m)} \tag{3.10}$$

Sinha and Tompa [2002] define a measure of significance $\rho_{max}(x)$, which measures the probability that the maximum $z-$score is at least x, if the input sequences were random. ρ_{max} is precomputed as follows. Random sequences of length l are generated by $\mathcal{M}$. YMF is run on these random sequences and the maximum $z-$score is recorded. This experiment is repeated 100 times and $\rho_{max}(x)$ is defined as the fraction of experiments that yield maximum $z-$score at least x.

Motifs for which $rho_{max}(z_s)$ is less than 0.1 are chosen.

YMF uses an HMM that is trained on yeast promoter sequences. Their web interface now has an HMM for *Arabidopsis thaliana* promoters but the major shortcoming of YMF is that it finds only short motifs in DNA sequences. As discussed earlier in this section, *Arabidopsis thaliana* motifs vary in length considerably. The motif model of YMF is a motif of length L with at most w spacers in the middle. This is not adequate to represent the more complex motifs in *Arabidopsis thaliana*.

Sift

Hudson and Quail [2003] are the first to develop an enumerative model called *Sift* that discovers regulatory motifs in *Arabidopsis thaliana* upstream regions. However, their method does not address the occurrence of motif combinations. Let G be a set of genes representing a regulon. Let Y be the set of all genes in the microarray from which G has been sampled. The foreground set of sequences is $\mathcal{U}(G)$ and the background set of sequences is $\mathcal{U}(Y)$. All upstream sequences are 2000 bases long sequences upstream of the translation start site.

The set of motifs M is defined as $\{m \mid m \in \Sigma_{DNA}^i,\ 1 \leq i \leq 10.\}$. Sift uses exact pattern matching to locate motifs. A 2-step filter is used to enumerate overrepresented motifs. The first filter ensures that the motifs are over-represented on the basis of their overall count by a chi-square test of independence and a Bonferroni correction. For a motif $m \in M$, let o_m be the number of occurrences of m in $\mathcal{U}(Y)$. Then, the expected number of occurrences of m in G is

$$e_m = \frac{o_m|G|}{|Y|} \tag{3.11}$$

and the chi-square score is

$$\chi^2 = \frac{(o_m - e_m)^2}{e_m} + \frac{([|G| - e_m] - [|G| - e_m])^2}{|G| - e_m} \tag{3.12}$$

The second filter ensures that the motifs are over-represented on a per-promoter basis. Let X be a random variable that represents the number of promoters in G that have at least 1 occurrence of m. X is assumed to follow a binomial distribution. Let π denote the probability of occurrence of m in each promoter. Y_m is the number of promoters in Y where m occurs, then

$$\pi \approx \frac{Y_m}{|Y|} \tag{3.13}$$

If X_m is the number of promoters in G where m occurs, the probability of observing X_m occurrences of m is

$$P_r[X = X_m] = \frac{(|G|)!}{X_m!(|G| - X_m)!}\pi_m^X(1-\pi)^{|G|-X_m}$$

Elements with a final p-value less than 10^{-5} are included in the result set.

3.2.2 Probabilistic Methods

Regulatory motifs are also detected using probabilistic methods where the input sequences are aligned and their local alignment scores are optimized to find conserved motifs based on the probability of finding each nucleotide at every position. Some algorithms also use a background model to

compensate for motifs that occur at high frequencies in both (foreground and background) sets, and eliminate them from the set of discoveries. Background models are also used to utilize genes not belonging to the co-regulated set for negative scoring. Lawrence et al. [1993] built an optimized local alignment model by iterative sampling, and this alignment also discovered patterns, repeats, and their optimal positions in the input sequences. Later, Thijs et al. [2002] improved the existing Gibbs sampling method for detecting overrepresented motifs by incorporating a higher order background model based on Markov chains and optimized them for the *Arabidopsis thaliana* and yeast genomes. A motif in which the average nucleotide composition differs significantly from the background is assigned a higher score. One of the drawbacks of alignment methods is that they do not differentiate between regulatory sequences specific to a co-regulated gene-set and core promoter motifs [Hudson and Quail, 2003]. These methods are computationally expensive and need systems with very high memory and CPU speeds to analyze a set of sequences that is not small. Such methods need to be trained on training data before they can be used. Most of these tools are trained and optimized for yeast or mammalian genomes and are therefore not appropriate for analyzing *Arabidopsis thaliana* promoters.

Probabilistic methods that have been employed include weight-matrix models of motifs, expectation maximization, and Gibbs sampling for motif detection. Weight matrices are created from multiple alignment of TFBSs and carry the probability of finding a particular base at each position of a TFBS. Weight matrices for most TFBSs can be obtained from the TRANSFAC database which mostly has PWMs for yeast, mammalian, and vertebrate genomes. However, this database still does not have weight matrices for most TFBSs in *Arabidopsis thaliana*. A number of algorithms ([Quandt et al., 1995, Schug and Overton, 1998, Kel et al., 2003]) have used weight matrices in motif detection. Algorithms using expectation maximization use maximum likelihood estimation to discover those values of the parameters of the model that maximize the likelihood of the data. In the first step they calculate the likelihood that a motif located at each position corresponds to the existing motif model. The second step is used to estimate the parameters that maximize the likelihood. Prominent tools that employ this method include MEME, BioProspector, AlignACE and LOGOS [Xing et al., 2004]. Gibbs sampling techniques have been used in Lawrence et al. [1993], Thijs et al. [2002], and Roth et al. [1998]. We will not go into further details of probabilistic methods in this work.

Chapter 4

The Regulatory Biclique problem

Binding of TFs to *c*is-elements is highly sequence specific and each *c*is-element binds only a subset of transcription factors. The mechanism by which promoter sequence differences are translated into different receptivity to specific transcription factors is not yet known but sequence specificity and conformational changes might be involved [Fickett and Hatzigeorgiou, 1997]. Binding of one or more TFs to corresponding *cis*-elements in a promoter may initiate, terminate, enhance or repress transcription. The presence or absence of a *cis*-element in the promoter of a gene determines whether a particular TF can bind to it. TFs are often large proteins or protein-complexes and this imposes geometrical and spatial constraints on the separation between *cis*-elements [Terai and Takagi, 2004]. The rate of transcription is also affected by the binding and interaction of more than one TF at the same time [Werner, 1999, Pilpel et al., 2001]. The combination of *cis*-elements present in a promoter is hypothesized to facilitate TF binding and influence gene expression. Often, the distance from the TATA box of the site where a TF binds is responsible for the control the TF has over gene transcription [Shen and Ho, 1995]. Therefore, transcriptional regulation is affected by the following factors:

- *Cis*-elements present
- *Cis*-element combinations present
- Distance between *cis*-elements
- Order of *cis*-elements

- Distance of a *cis*-element from the transcription start site

We call a set of regulatory elements co-occurring in a promoter a motifset. The factors of order and separation of motifs within the motifset will not be addressed in this work. However, the occurrence of motifsets will be analyzed in detail. Identification of statistically significant motifsets based on sequence analysis alone will generate a large number of motifsets that are statistically significant. However, the presence of not all of these motifsets influence gene expression. Specifically, in *Arabidopsis thaliana* some *cis*-elements, such as those related to water-stress are present in the promoters of a large proportion of genes in the genome. The motifs for these elements usually appear as significant motifsets, whereas their co-occurrence cannot be biologically justified as responsible for stress responses other than water stress.

A set of genes is referred to as a geneset. A geneset that is interesting to a biologist does not necessary consist of genes, all of which are regulated exactly in the same manner by synonymous TFs. However, if the genes in the geneset are chosen based on some common biological criteria, it is possible that some genes in the geneset share common regulatory modules or motifsets whose presence actually influences expression. **A set of genes and motifs, such that every motif is present in the promoter of every gene in the set, will be referred to as an *itemset***. An itemset I consists of a geneset G and a motifset M and is represented as $I_i = \langle G_i, M_i \rangle$. Therefore, given a set of genes and a set of regulatory motifs, we can identify itemsets that are over-represented, statistically significant by sequence analysis, and also consist of genes that are co-expressed with respect to functional data such as gene expression under a specified set of treatments. A motifset whose presence renders gene expression is said to be synergistic. An itemset with a synergistic motifset is biologically pointed. We call this the *Regulatory Biclique problem* since it relates the presence of a motifset to a typical expression pattern in a geneset. The Regulatory Biclique problem can be computationally formulated as in Section 4.1.

4.1 Computational Problem Formulation and Solution

INSTANCE: A set **G** of genes, a set **P** of patterns, and a set **T** of treatments.

SOLUTION:

1. A set of itemsets I, $I_i = \langle G_i, P_i \rangle$, $1 \leq i \leq |I|$, $G_i \subseteq \mathbf{G}$, and $P_i \subseteq \mathbf{P}$.

   (1) For each itemset I_i, $1 \leq i \leq |I|$, the SAV statistic:

   $$S_i \ = \ \sum\nolimits_{1 \leq p < q \leq |G_i|} |\rho_{g_p, g_q}|, \tag{4.1}$$

   where $g_p, g_q \in G_i$.

   (2) For each SAV statistic S_i of itemset I_i, the tail probability:

   $$1 - F_X(S_i), \tag{4.2}$$

   where X is the random variable representing the SAV statistic and $F_X(S_i)$ is precomputed by random sampling from the genome for a geneset of size $|G_i|$.

   (3) For each itemset I_i, the hypergeometric tail probability of the occurrence of P_i in G_i:

   $$P_r[Y > |G_i|] = \mathcal{H}_{tail}(N, \mathcal{C}_{P_i}, |\mathbf{G}|, |G_i|) \tag{4.3}$$

   where Y is the random variable representing the number of occurrences of P_i, N is the total number of genes in the AT genome, and $\mathcal{C}_{P_i}$ is the number of genes in the genome such that $P_i \sqsubset \mathcal{C}_{P_i}$.

2. A set of relationships

   $$\mathcal{R} = \{\mathcal{R}_i | \mathcal{R}_i = \langle G_i, P_i, Q_i \rangle\}, \tag{4.4}$$

   where, $1 \leq i \leq |I|$, $G_i \subseteq G$, $P_i \subseteq P$, and

   $$\mathcal{Q}_i = \mathcal{H}_{tail}(N, \mathcal{C}_{P_i}, |G|, |G_i|) \times (1 - F_X(S_i)). \tag{4.5}$$

4.2 Motivation

A large number of programs have been developed for discovering *cis*-regulatory modules in yeast. These methods have used different approaches to recognizing co-occurring *cis*-elements, such as

experimental data, word-pair analysis of all putative motifs, and Monte-Carlo screening of motifs, which are described in more detail in Section 4.4. Pilpel et al. [2001], Kato et al. [2004] and Chiang et al. [2003] identified synergistic motif combinations in yeast. Identifying co-regulated genes with common regulatory motifs has been very successful in yeast [Riechmann, 2002]. One possible reason for this could be that yeast, being single-celled, have a simpler transcriptional machinery than multicellular eukaryotes. The yeast genome is known to have 6000 genes, including genes for about 50 transcription factor proteins. Yeast cells grow quickly and have been widely studied. The transcriptional mechanisms in yeast are well understood and there is enough biological data about yeast upon which to base computational findings. In a higher eukaryote like *Arabidopsis thaliana*, the gene-density is much higher (approximately $28,000$). Approximately 1500 *Arabidopsis thaliana* transcription factors have been identified. It is possible that more TFs will be identified in future. While a number of databases exist that contain details of all TFBSs in yeast, not all TFBSs in *Arabidopsis thaliana* are known and documented. Only a fraction of TFBSs in *Arabidopsis thaliana* have documented consensus sequences or position weight matrices. The absence of enough biological information in the case of *Arabidopsis thaliana* makes the validation of all promoter discovery tools difficult.

4.3 Identifying Putative Co-Regulated Genes

Genes that are regulated by common TFs, share the same regulatory motifs. Orthologous genes are defined as having the same function across species and hence, share similar regulatory structures, whereas it is less likely that paralogs share similar regulatory structures [Rombauts et al., 2003].

Clustering of gene expression data is frequently used for identifying putative genes with common regulatory motifs. A vector that stores expression levels of a gene for a given set of experiments is the *expression vector* of that gene. Genes that display similar transcriptional responses in a set of experiments exhibit similar expression vectors and are said to be co-regulated. The promoters of co-regulated genes are hypothesized to contain the same *cis*-elements.

Gene annotations and GO categories are yet another way of identifying genes that share a common function an, possibly, common regulatory motifs.

4.4 Combinatorial Analysis Programs

Programs that analyze occurrences of individual motifs, assume that each *cis*-element occurs independently of the others, whereas initiation of transcription by cooperative binding is the actual biological phenomenon. In reality, co-occurrence of TFBSs is tightly coupled. Pilpel et al. [2001] identify regulatory networks in yeast by building a database of known and putative yeast TFBSs and identifying synergistic motif combinations based on the expression coherence score of each gene set having a pair of motifs. The motif-association maps that they generate are highly connected, suggesting that transcription factors work in combinations to render different expression patterns. This indicates that if two or more motif occurrences are tightly coupled, then individual occurrences of each of these motifs, will not contribute to meaningful biological function and are better eliminated as false positives. Enumerative approaches that calculate the significance of individual motifs might not be able to strongly correlate promoter patterns with expression patterns based on occurrences of individual motifs.

The process used by Pilpel et al. [2001] to identify synergistic motifsets is summarized below. The input sequence set U is the set of upstream regions of yeast genes in MIPS functional categories. Fig 4.4 explains the curation of a data set M_{final} of known and putative yeast regulatory motifs and the assignment of motifs to upstream regions.

Figure 4.1: Curation of yeast regulatory motifs in Pilpel et al. [2001].

1. Apply AlignACE to U to get a set M_1 of 819 motif matrices.
2. Hierarchically cluster all motifs in M_1 into a set of clusters C.
3. Compute the set $M_2 = \{m_i | m_i$ is the motif with the highest group specificity score in cluster$\}C_i \in C, 1 \leq i \leq |C|$.
4. Curate the set set M_K of known yeast regulatory motifs from literature and the SCPD database.
5. Compute $M_{final} = M_2 \cup M_K$.
6. For each $m_i \in M_{final}$, $1 \leq i \leq |M_{final}|$
   (1) $U(m_i)$ is the set of upstream regions used to derive m_i, $U(m_i) \subseteq U$.
   (2) Define $s(m_i, u)$ as the scanACE score of motif m_i on upstream region $u \in U(m_i)$.
   (3) $\mu(m_i)$ is the mean and $\sigma(m_i)$ is the standard deviation of the scanACE scores of m_i for all upstream regions in $U(m_i)$.
   (4) Assign $m_i \sqsubset u$, $u \in U(m_i)$, if $s(m_i, u) > \mu(m_i) - 2 \times \sigma(m_i)$.
   (5) If $|\{u | m_i \sqsubset u\}| > 300$, consider m_i to be present only in the top 300 scoring upstream regions.

   End

The *expression coherence score (EC)* for a motifset $M \subseteq M_{final}$ is defined as follows. Let G be a set of genes such that $M \sqsubset \mathcal{U}(G)$. Let e_g denote the mean and variance normalized expression profile of gene g. For every gene pair $(g_1, g_2) \in G$ the Euclidean distance $\|e_{g_1} - e_{g_2}\|$ is computed. D is a pre-computed threshold Euclidean distance. Let $n_{G,M}$ be the number of gene-pairs in G whose Euclidean distance is smaller than D. Then the EC of motifset M in geneset G is defined as

$$EC_{M,G} = \frac{n_{G,M}}{\binom{|G|}{2}} \tag{4.6}$$

To calculate D, 100 genes are randomly sampled from the genome and the Euclidean distances for all $\binom{100}{2}$ gene pairs is computed. D is defined as the lowest value in the fifth percentile of the distribution of these distances.

Pilpel et al. [2001] calculate EC scores for genes containing 2 motifs in their upstream regions, considering only motif pairs that are present in at least 10 genes. Consider a motifset $M = m_1, m_2$ with 2 motifs. Let G be the geneset such that $M \sqsubset \mathcal{U}(G)$. Let G_1 and G_2 be genesets such that $m_1 \sqsubset \mathcal{U}(G_1)$ and $m_2 \sqsubset \mathcal{U}(G_2)$, respectively. Then the motifset M is defined as *synergistic* if $EC_{M,G}$ is significantly greater than $max(EC_{\{m_1\},G_1}, EC_{\{m_2\},G_2})$. The test of hypothesis for this is done using a Monte Carlo procedure.

For a given expression experiment, n motifs are chosen from synergistic motif combinations. Each gene in the yeast genome is then assigned a binary signature of length n, placing a 1 at the i^{th} position if a gene contains the i^{th} motif, and 0 otherwise. This generates 2^n genesets. Such a geneset is called a GMC (genes defined by motif combinations). All genes in a GMC share the same motif signature. For each GMC, the EC score and an average expression profile is generated. The Pearson correlation coefficients between average expression profiles of all pairs of GMCs is computed. These correlations are used to compute a dendrogram.

Another output of this system are motif synergy maps drawn by the GEM algorithm on the Brown University GeomNet server. The input to the algorithm is a set of synergistic motif pairs where at least one of the motifs is known. The algorithm lays out motifs in a plane and edges between motifs indicate motif pairs whose synergy score is below a given threshold t. t is set to $\frac{1}{(\#\ of\ motifs)\times(\#\ of\ known\ regulatory\ motifs)}$.

XcisClique differs from the system developed by Pilpel et al. [2001] (we will call this IRNCAPE from the initials in the paper title) in the following ways. Regulatory motifs in IRNCAPE are modeled as weight matrices whereas, in XcisClique regulatory motifs are modeled as Perl regular

expressions. While 37 regulatory motifs for yeast had been identified at the time that IRNCAPE was built, the number of known TFBSs for *Arabidopsis thaliana* is close to 120 and almost all of them have been experimentally determined. Unlike IRNCAPE, which restricts the number of upstream regions where a motif is present to 300, XcisClique has no restrictions on the number of promoters a regulatory element is present in. Presence of a regulatory element in a promoter is determined by exact pattern matching of Perl regular expressions. IRNCAPE allows construction of GMCs only over a single expression experiment at a time. In XcisClique, 9 abiotic stress experiments are available and itemsets can be analyzed over any combination of these treatments. In IRNCAPE, a motifset is searched in all genes of the genome and GMCs are determined from the results, but XcisClique identifies itemsets using the input set of genes. Most importantly, no steps in XcisClique involve imposing cutoffs to reduce search space. Instead, search space is reduced by adding biological context at the outset of an analysis while choosing the input genes, *cis*-elements, and treatments.

Kato et al. [2004] integrate chromatin-immunoprecipitation (ChIP) data with combinatorial motif analysis to identify over-represented motif combinations. Subsequently, they identify motif combinations whose genes have coherent expression patterns. Their analysis directly relates co-occurrence of motifs with gene function. However, this analysis was done with yeast genes for which ChIP and microarray expression data are readily available. It is difficult to do such an analysis with *Arabidopsis thaliana* genes for lack of ChIP data.

Chiang et al. [2003] associated conserved word pairs (with respect to phylogeny and spatial separation) in yeast with gene expression. These words were regulatory templates (not exactly motifs), hexamers that are jointly conserved at non-random distances. Their approach only relates pairs of regulatory words but this is a step towards determining if order is important among regulatory motifs.

Gupta and Liu [2005] identify CRMs (*cis*-regulatory modules) eukaryotic genomes (*B. subtilis*, *D. melanogaster*, *H. sapiens*, and *M. musculus*). But their method needs PWMs from TRANSFAC [Wingender et al., 1997] and JASPAR [Sandelin et al., 2004], and so is not applicable to *Arabidopsis thaliana*.

A common drawback of probabilistic methods is that they consider n-mers only (typically hexamers) in discovering significant motifs. This discovers only regulatory templates (which are hexamers separated by a few spaces) and not the actual motifs. TFBSs in *Arabidopsis thaliana* widely vary in length, for instance, the heat shock element is 13 nucleotides long and the `ACGTATERD1` is 4 nu-

cleotides long. Also, most tools generate a large number of false positives in their discoveries, because of noise, the model, calculation of the background model, and modeling parameters [Rombauts et al., 2003]. The optimal model parameters to use are dependent upon the organism, the tissue type, the regulatory process and TFBSs in question.

4.5 Solution to The Regulatory Biclique problem

In this thesis, the Regulatory Biclique problem is solved via 8 major steps. The first 4 steps are directed at setting up the local database with XcisCliquecentric sequence and gene expression data. Subsequent steps execute an analysis for a given input set of genes, *cis*-elements, and treatments.

First, annotated sequence data for *Arabidopsis thaliana* genes and their upstream regions are stored in a local database. Second, annotated *Arabidopsis thaliana cis*-elements from multiple sources are curated and stored. Third, matches of all *Arabidopsis thaliana cis*-elements in the promoters of all *Arabidopsis thaliana* genes are computed and stored in the database. In the fourth step, gene expression data for $22,814$ AT genes are processed into gene-specific expression vectors. Central values of the SAV statistic and their regression fits for genesets of various sizes and the set of all treatments are precomputed and stored.

In the fifth step, promoters of an input geneset are scanned for the input set of *cis*-elements. Maximal pairs of motifsets and genesets are then clustered into itemsets by the Apriori algorithm (Ref. Section 7.4). Sixth, in each itemset, the distribution of motifsets is treated as hypergeometric and a p-value is generated corresponding to the number of occurrences of the motifset in the geneset. In the seventh step, pairwise Spearman correlation coefficients between all pairs of genes in each itemset are computed over the input set of treatments and stored in the database. For the geneset in each itemset, the SAV statistic is computed and its tail probability under the normal distribution is stored as a p-value. In the eight and final step, the product of the p-values from sequence analysis and expression data analysis is stored and reported as the final p-value for each itemset. This is an indicator of how significant the presence of the motifset is in the geneset, as well as the correlation between gene expression vectors of genes in the geneset. This solution is database centric and specific to *Arabidopsis thaliana* at this point.

The novelty of XcisClique lies in the fact that this is the only tool of its kind for *Arabidopsis thaliana* to provide an integrated infrastructure for annotated genome data, annotated *cis*-element

data and gene expression data. XcisClique detects statistically overrepresented itemsets, as existing tools for other species identify motifsets as described above, but XcisClique also evaluates each itemset with respect to gene expression data. This gives an indication of the importance of co-occurrence of a set of regulatory elements in a geneset with respect to transcriptional response. The p-value of each itemset, is therefore a determinant of biological significance as well.

Following is an example of an itemset from an analysis in XcisClique. The input set of genes is set of 17 genes involved in stress, pathogenicity and secondary metabolites in *Arabidopsis thaliana* [Gepstein et al., 2003]. Expression data for 15 of the 17 genes is available in the POPS database (Chapter 5). Promoters of length 1200 for the input geneset were scanned for the set of all *Arabidopsis thaliana* *cis*-elements. Expression data were correlated over a set of 7 treatments (Cold, Heat, Drought, Osmotic, Oxidative, Salt, UVB) in shoots. The 32^{th} itemset I_{38} identified by the Apriori algorithm has a p-value of $3.955E-03$ from sequence analysis and a p-value of $1.236E-02$ from expression data analysis. $I_{38} = \langle G_{38}, M_{38} \rangle$ where G_{38} is {At1g09500, At1g73160, At4g37990, At4g39090, At5g59310} and M_{38} is {ARFAT, DPBFCOREDCDC, GT1CONSENSUS, MYCCONSENSUSAT, RAV1AAT, ZAT12-down}.

XcisClique uses only known *Arabidopsis thaliana* motifs curated from various sources. The system does not have a motif discovery process integrated into it. This ensures that the search space for patterns is limited, not confined by motif lengths, and consists of well-defined, annotated motifs. The biologist has the choice of selecting a subset of relevant motifs, and this makes one of the three inputs (*cis*-elements) biologically directed. The genes being analyzed are yet another input to the system by the biologist and these genes are chosen on a biological basis. The treatment sets, over which expression data is to be considered, are the third input to the system and can also be specified by the biologist over a range of 9 abiotic stress treatments. The integration of biological knowledge into XcisClique inputs, greatly reduces the final search space and produces more biologically relevant results.

XcisClique is scalable to more numerous motifs and treatments. In the presence of appropriate sequence and gene expression data, the system can accommodate any organism, and is generic in that sense. Combinations of motifs can be viewed through the PHP-based viewer *MotifSee* that is included with the system. In the presence of processed gene expression data, XcisClique can be used for any organism. The system has been verified with biological data from *Arabidopsis thaliana*.

Chapter 5

Data Sources and Databases

This chapter describes the data sources and database organization for XcisClique.

5.1 Sequence Data

In XcisClique, analysis of *cis*-elements in promoters has been done with *Arabidopsis thaliana* as a model organism. Hence, it is expedient to have a local database of Arabidopsis genes, proteins, and promoters. This database is called "POPS" and is currently hosted on `http://expresso.cs.vt.edu`. POPS stands for "Plant Ortholog Paralog Species" and is built on a postgres platform. POPS is populated using Perl scripts that use *Entrez* programming utilities called "e-utilities". *Entrez* is the search engine for all NCBI databases, and e-utilities provide access to *Entrez* data via web services from Perl and Python programming environments. These programming utilities are extremely useful in constructing pipelines of search results. *Entrez* supports the following e-utilities programs: EInfo, ESearch, EPost, ESummary, EFetch, ELink, and EGQuery on about 26 databases (For example, Nucleotide, Protein, Genome, Gene, Structure, Homologene, SNP, Pubmed Central, OMIM, GEO Datasets, CDD, Taxonomy, Pubchem Compounds). Each of these queries can be tailored to retrieve customized results, and it is possible to pass results from one query to another. Further information about e-utilities can be found in Sayers and Wheeler [2003] or at `http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html`. The POPS database has been created by retrieving database fields selectively to create a schema tailored to the needs of XcisClique. The POPS database currently consists of the 15 tables in Figure 5.1.

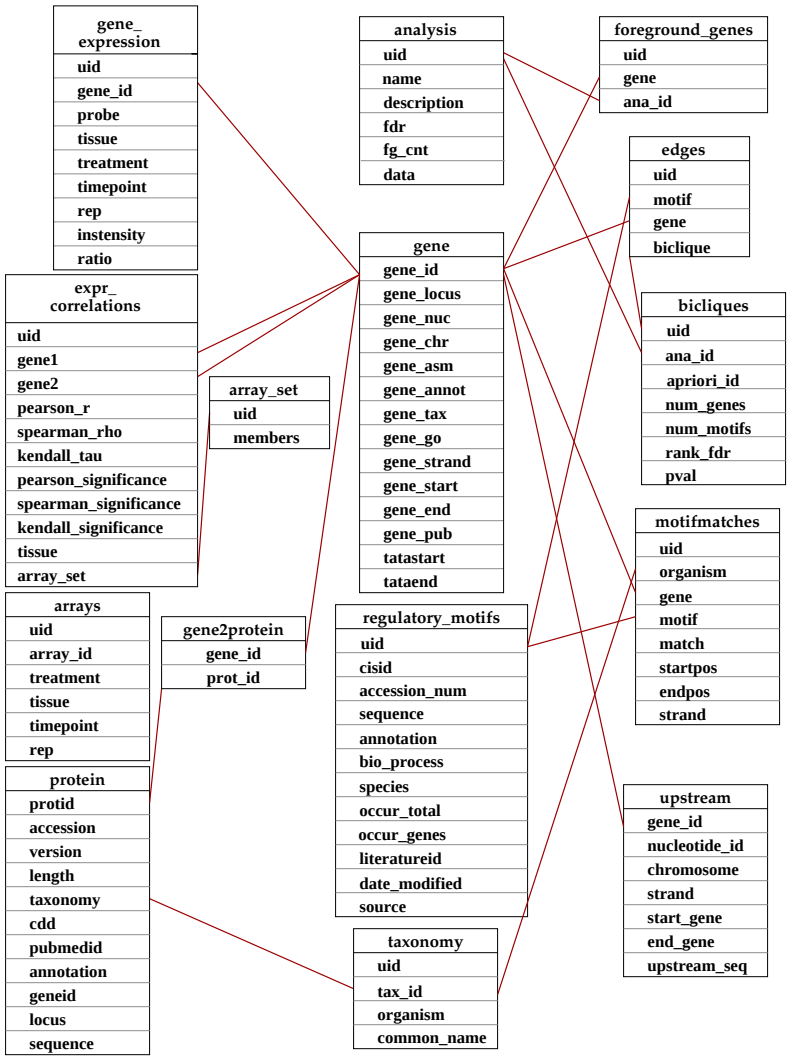


Figure 5.1: Entity-Relationship diagram of the POPS database

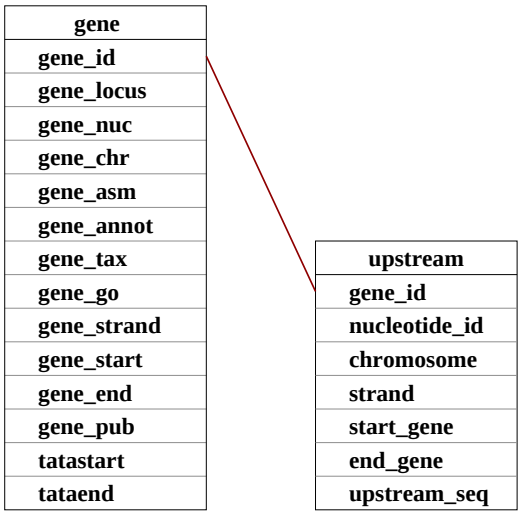


Figure 5.2: `gene` and `upstream` tables in the POPS database

The `gene` table is the central table in the schema. The `upstream` table is linked only to the `gene` table. See Figure 5.2. The `gene` table contains 31,000 distinct gene records. Since each of these records has a unique identifier at NCBI (field `gene_id`), and 28,000 *Arabidopsis thaliana* genes are known, the numbers indicate that for some genes there might be more than one gene record at NCBI owing to significance difference in the sequences submitted for that gene. The `gene_id` field maps to NCBI's gene identifiers. The accession number (field `gene_locus`), annotation (field `gene_locus`), chromosome (field `gene_chr`), nucleotide id (field `gene_nuc`), gene assembly id (field `gene_asm`), taxonomy (field `gene_tax`), GO category (field `gene_go`), pubmed id of related literature (field `gene_pub`), gene-annotation (field `gene_annot`), and start (field `gene_start`) and end (field `gene_end`) positions of the gene on the nucleotide on which it is located, are also stored in this table. For each gene in the local table, the upstream region is extracted and stored in the `upstream` table along with other fields as indicated in Figure 5.2. Upstream sequences of length 2000 bases, stored in this table are used for analysis by XcisClique. It is related to the `gene` table via the `gene_id` field. Other fields in this table are redundant with respect to the `gene` table.

Regulatory motifs and their matches on upstream regions of genes are stored in tables `regulatory_motifs` and `motifmatches` as shown in Figure 5.3. In the `regulatory_motifs` table, the fields `cisid`, `accession_num`, `sequence`, `annotation`, `bio_process`, `species`, `literatureid`, and `date_modified` are self-explanatory and represent information from PLACE. `occur_total` and

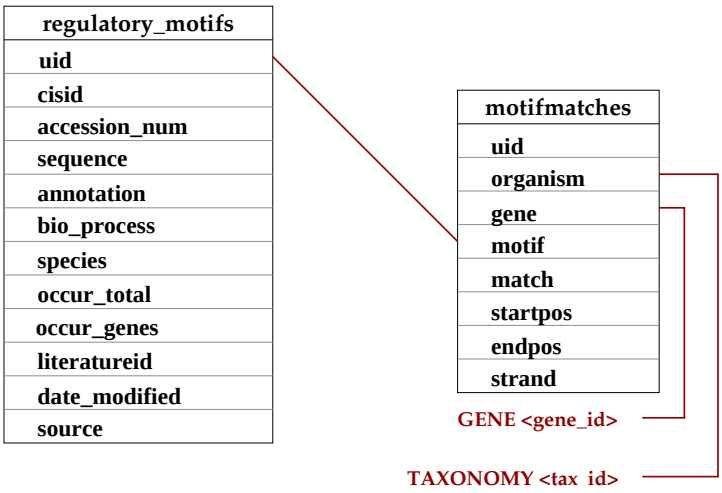


Figure 5.3: `regulatory_motifs` and `motifmatches` tables in the POPS database

`occur_genes` are XcisClique computed fields that represent the total number of occurrences of the RE in the entire *Arabidopsis thaliana* set of promoters, and the total number of genes in the promoters of which the RE occurs in *Arabidopsis thaliana*, respectively. The percentage of gene promoters in the entire genome, that contain a particular motif, is an indicator of the rarity of that motif.

Figure 5.4 shows the tables that are dynamically populated when an analysis is run using XcisClique. For every new analysis performed by XcisClique, the particulars (name (field `name`), description (field `description`), false discovery rate (field `fdr`), foreground count (field `fg_cnt`), and date (field `date`)) of the analysis are stored in the `analysis` table. All the genes in the geneset being analyzed are stored in the `foreground_genes` table, which relates to the `gene` table via the `gene` field. All the itemsets identified during an analysis are stored in the `bicliques` table that stores the identifier of the analysis with which the itemset is associated (field `ana_id`), the apriori generated identification for the itemset (field `apriori_id`), number of genes in the itemset (field `num_genes`), number of motifs in the itemset (field `num_motifs`), rank of the itemset when all itemsets are arranged in increasing order of their p-values (field `rank_fdr`), and the p-value of the itemset (field `pval`). The `edges` table stores gene-motif pairs for each itemset. The `gene` field links to the `gene` table (field `gene_id`). The `motif` field links to the `regulatory_motifs` table (field `uid`).

Figure 5.5 shows the 6 tables used to store gene expression data. The `gene_expression` table that stores *Arabidopsis thaliana* gene expression data from the Nottingham database. Pearson (field

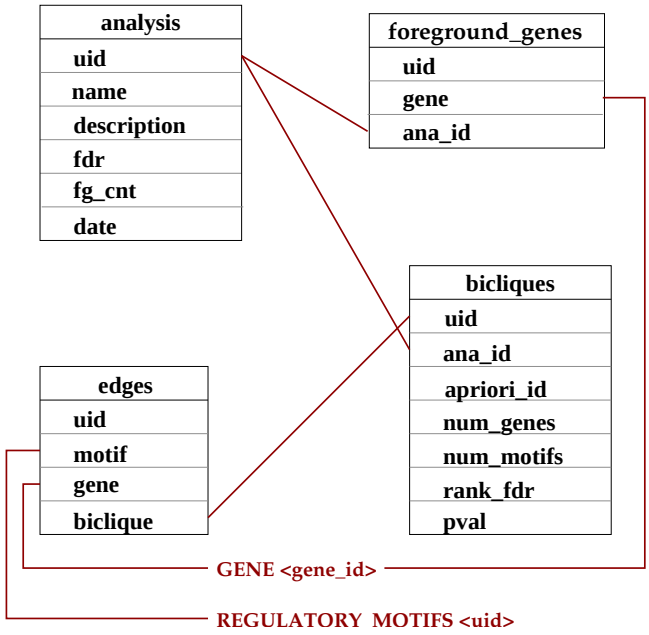


Figure 5.4: `analysis`, `foreground_genes`, `bicliques`, and `edges` tables in the *pops* database

`pearson_r`), Spearman (field `spearman_rho`), and Kendall (field `kendall_tau`) correlation coefficients between pairs of genes, and the p-values for these coefficients (fields `pearson_significance`, `spearman_significance`, and `kendall_significance`) are in the `expr_correlations` table. Details of how these fields are computed are explained in Figures 7.5.1 and 7.5.2 and definitions of these correlations are in Section 6.1. The `correlation_cv_dist` table stores the central values of the mean and standard deviations of spearman correlation coefficients, computed for various sample sizes of randomly chosen genes. This table also stores the mean and standard deviations of the S (SAV) (Section 7.5.2) statistic for various sample sizes of randomly chosen genes. Distributions of S for different geneset sizes are stored in the table `s_distribution`. The entries in this table are used to evaluate an itemset in terms of the probability of occurrence of a statistic of that itemset, using a distribution of the statistic generated from random sampling. The `regression_parameters` table stores linear regression fits corresponding to statistics (or their log transformations). The `arrays` table is used to store all microarrays and the `array_set` table stores combinations of these microarrays used for analyses. All field names in these tables are self-explanatory.

The `protein`, `gene2protein`, and `taxonomy` tables are not used in XcisClique analysis. These

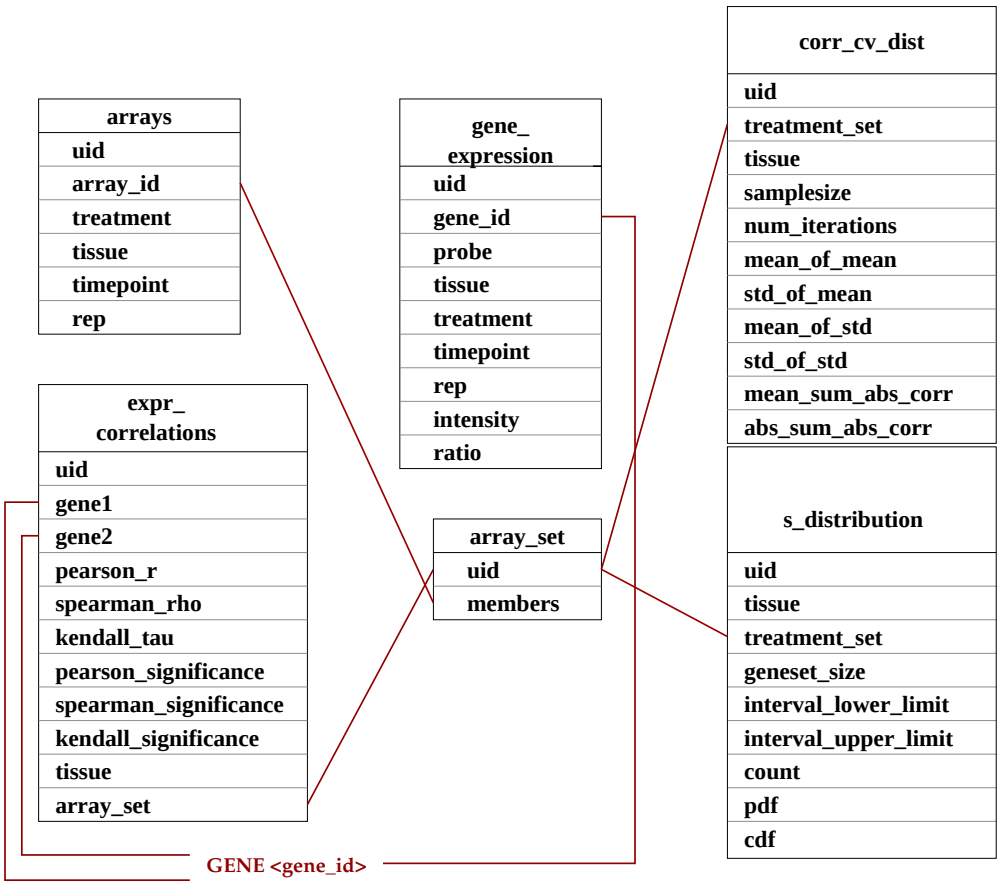


Figure 5.5: `expr_correlations`, `correlation_cv_dist`, `s_distribution`, and `gene_expression` tables in the POPS database

are present to lend a complete structure to the database and are illustrated in Figure 5.6.

XcisClique is designed to dynamically populate database tables with results of analyses. Pre-populated tables include `gene`, `protein`, `upstream`, `regulatory_motifs`, `arrays`, `array_set`, `treatments`, `treatment_sets`, and `gene_expression`. Appendix C contains the Perl scripts used to populate these tables. Tables that are populated during analyses are `analysis`, `foreground_genes`, `bicliques`, `edges`, `expr_correlations`, `correlation_cv_dist`, `s_distribution`, and `regression_parameters`.

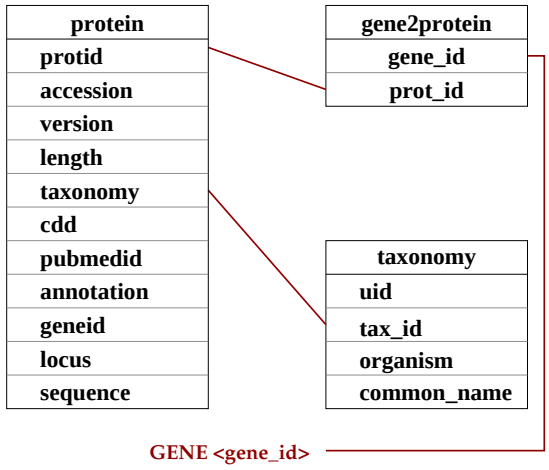


Figure 5.6: `protein`, `gene2protein`, and `taxonomy` tables in the *pops* database

5.2 Gene Expression Data

Expression data for the *Arabidopsis thaliana* transcriptome was retrieved from NASC arrays in the Nottingham database (`http://www.arabidopsis.org.uk/`). The slides are Affymetrix ATH1 Arabidopsis Genome Arrays having 22,814 genes. These data have been generated as part of the AtGenExpress project funded by Das von der DFG geförderte AFGN (Arabidopsis Functional Genomics), that aims at providing the Arabidopsis community with access to a large set of Affymetrix microarray data. This project has generated expression data from 80 biologically different samples. The available data have been analyzed using the Affymetrix Microarray Analysis Suite 5.0 with the Affymetrix MAS 5.0 Scaling Protocol. The top 2% and bottom 2% of signal intensities were excluded, then the mean was calculated. The original signal values were scaled such that the mean was made equal to 100. This data is normalized.

We selected 272 slides for abiotic stress experiments. There are 9 abiotic stress experiments as follows: *Salt, Drought, Genotoxic, Oxidative, UV-B, Wounding, Heat, Cold and Osmotic*. Expression data for each of these is available over a series of time points ($0.25h$, $0.5h$, $1h$, $3h$, $6h$, $12h$, $24h$) with 2 biological replicates per time-point. Control slides also exist for each of these time points. Depending on time-sets that were common to all time points and controls, we have selected 5 time points over which expression values are considered: $0.5h$, $1h$, $3h$, $6h$ and $12h$. All expression data are intensity values. 136 of the 272 slides contain genes from shoots and the other 136 slides contain

genes from roots. All gene expression data are stored in the `gene_expression`, `arrays`, `array_set`, `treatments`, `treatment_sets`, `expr_correlations` and `correlation_cv_dist` tables.

5.3 *Cis*-Element Data Sources

PLACE, which is a database of plant *cis*-acting regulatory DNA elements [Higo et al., 1999], is our primary source for *cis*-regulatory element data. These have been compiled from previously published reports and covers vascular plants only. Their variations across other genes or in plant species are also reported along with literature references. *cis*-elements from PLACE have been stored in the `regulatory_elements` table. XcisClique uses the subset of *Arabidopsis thaliana cis*-elements present in this table. *Arabidopsis thaliana* motifs have also been curated from various sources in literature when required by an analysis. All *cis*-elements used in XcisClique are listed in Appendix A.

Chapter 6

Probabilistic Methods

In XcisClique, analysis of gene expression data and annotated genomes employs a number of probabilistic techniques. Correlation methods (Section 6.1) are used extensively in the analysis of gene expression data. Section 6.2 describes the hypergeometric distribution, which is used to discover over-represented motifs.

6.1 Correlation Methods

Correlation is a bivariate measure of association or strength of the association between two vector variables.

6.1.1 Pearson's Correlation Coefficient

Pearson's correlation coefficient is a statistic that estimates the correlation of a pair of random variables. It is also called the Pearson product-moment correlation coefficient, denoted by r, and is a measure of how well the relation between two vectors can be measured by a linear equation. Let x and y be vectors of length n. The Pearson correlation coefficient of x and y, $r(x, y)$ is computed by dividing the covariance between the two vectors by the product of their standard deviations:

$$r(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}. \tag{6.1}$$

where x_i indicates the i^{th} element of x and y_i indicates the i^{th} element of y. From equations 2.19 and 2.5,

$$r = \frac{\sum \mathcal{Z}(x)\mathcal{Z}(y)}{N-1}. \tag{6.2}$$

The Pearson correlation coefficient may take any value between -1 and $+1$. The sign of the correlation coefficient shows the direction of the relationship. A value of 1 indicates that the relationship between x and y can be perfectly described by a linear equation for a line with a positive slope. A score of -1 indicates a line with a negative slope. A 0 indicates that there is no linear relationship between the two variables. This linear equation can be found by linear regression. The square of r is called the coefficient of determination and measures the strength of association between X and Y. For example, if r is 0.95, then 90.25% of the variance in Y is explained by changes in X.

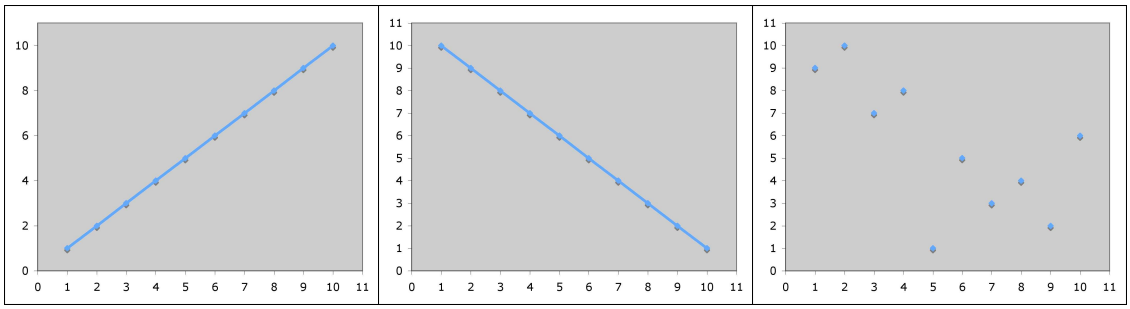
XcisClique calculates the Pearson correlation between gene-pairs in order to correlate expression data. However, the values of r are not used in further calculations because r is a parametric statistic, i.e., during statistical hypothesis testing, the distribution of the variables is assumed to belong to a known parametrized probability distribution, in this case, the normal distribution. The distribution of fold change of gene expression in response to various treatments is not known, and so, r cannot be used for further analysis and testing the significance of the correlation coefficient. When the underlying distributions of the variables are unknown, non-parametric methods like Spearman's ρ are more appropriate.

6.1.2 Spearman's Rank Correlation

Spearman's Rank Correlation [Sheskin, 2000], named for Charles Spearman and denoted by the Greek letter ρ, is a non-parametric measure of correlation and assesses how well an arbitrary monotonic function can describe the relationship between two variables, without making any assumptions about the frequency distribution of the variables. It does not require that the relationship between the variables be linear or the variables be measured on interval scales. It is used for variables measured at the ordinal level, i.e., variables that denote the position of an element in a sequence.

In principle, ρ is a special case of the Pearson product-moment coefficient in which data are converted into ranks before calculating the coefficient. In practice, however, a simpler procedure is normally used to calculate ρ. The raw scores are converted into ranks, and the differences d between the ranks of each observation on the two variables are calculated. We will use the following notation in explaining the calculation of the Spearman correlation coefficient. If x is a vector, the x^s is the

Table 6.1: Types of Spearman correlation coefficients, (i) Perfect positive correlation (+1), (ii) Perfect negative correlation (−1), (iii) No trend (0).



(i) (ii) (iii)

sorted vector of x and x^r is the rank vector of x. Let x and y be vectors of length n between which the Spearman correlation is to be computed. x^r is computed as follows. If $x^s_{j_1} = x^s_{j_1+1} = \ldots = x^s_{j_2}$ and maximal, then, for $j_1 \leq i \leq j_2$,

$$x^r[i] = \frac{j_1 + j_2}{2}$$

y^r is computed from y in exactly the same manner. The difference vector d is

$$d_i = |x^r_i \ - \ y^r_i|, \ 1 \leq i \leq n. \tag{6.3}$$

where d_i denotes the i^{th} element of d, x^r_i denotes the i^{th} element of x^r, and y^r_i denotes the i^{th} element of y^r. The Spearman Rank Correlation Coefficient between x and y is

$$\rho(x, y) = 1 - \frac{6(d(x, y))^2}{n(n^2 - 1)}. \tag{6.4}$$

Meaning of Spearman's Rank Correlation Coefficient

Spearman's ρ has a value ranging from -1 to 1. The closer this value is to -1 or 1, the stronger the likely correlation. 1 indicates a perfect positive correlation and -1 indicates a perfect negative correlation. These trends are represented by Figure 6.1.2. To test whether this value is significantly different from zero, the observed value can be compared with published tables with various levels of significance. For sample sizes above 20, the variable

$$\rho = \frac{6 \sum d^2}{n(n^2 - 1)} \tag{6.5}$$

has a Student's t-distribution. For large samples, the Student's t-distribution approaches the normal distribution. So, the distribution of ρ can be approximated as normal (since ρ is a discrete random variable, it cannot have a truly normal distribution that is continuous). Assuming normal distribution of ρ, the mean of the distribution of ρ, $\mu(\rho) \approx 0$ and the standard deviation is

$$\sigma(\rho) = \sqrt{\frac{n^2(n-1)(n+1)^2}{36}} \tag{6.6}$$

The number of degrees of freedom to be used is $(n-2)$ [Sheskin, 2000]. We use ρ to correlate gene expression vectors of size 45. At 43 degrees of freedom and ρ equal to 0.54 or more the likelihood of the correlation occurring by chance is very small (0.1%). Spearman's ρ is more reliable for sample sizes between 7 and 30 as compared with sample sizes out of these limits.

6.2 The Hypergeometric Distribution

We defined the hypergeometric distribution p-value in Section 2.1.3. This section describes the hypergeometric distribution in detail. The hypergeometric model assumes that in a set of n trials, there are two possible outcomes. Because sampling without replacement is assumed, the outcome of each trial will be dependent upon the outcomes of the earlier trials. The model is, therefore, similar to the model for the binomial distribution except for sampling without replacement [Sheskin, 2000]. So, the probability of obtaining an outcome in a given category will change from one trial to another; the value of the probabilities will be a function of the number of potential observations in each category that are still available for selection.

Consider a population of N objects, with C special objects. Figure 6.1 illustrates sampling from a population using the hypergeometric model. Let X represent the number of special objects that occur in a sample of size n chosen from this population. Then, the probability of X being exactly equal to c is

$$P[X = c] \;=\; \mathcal{H}(N, C, n, c) = \frac{\binom{n}{c} * \binom{N-n}{C-c}}{\binom{N}{C}} \tag{6.7}$$

Let us define the property as the "presence of a motifset M in the promoter" and the population as all genes in the *Arabidopsis thaliana* genome. Then $\mathcal{C}_M$ genes in the genome have the motifset (Section 2.1.2). If we sample n genes at random from the population, then equation 6.7 gives the probability of finding exactly c genes with the motifset in their promoters. So, the probability of

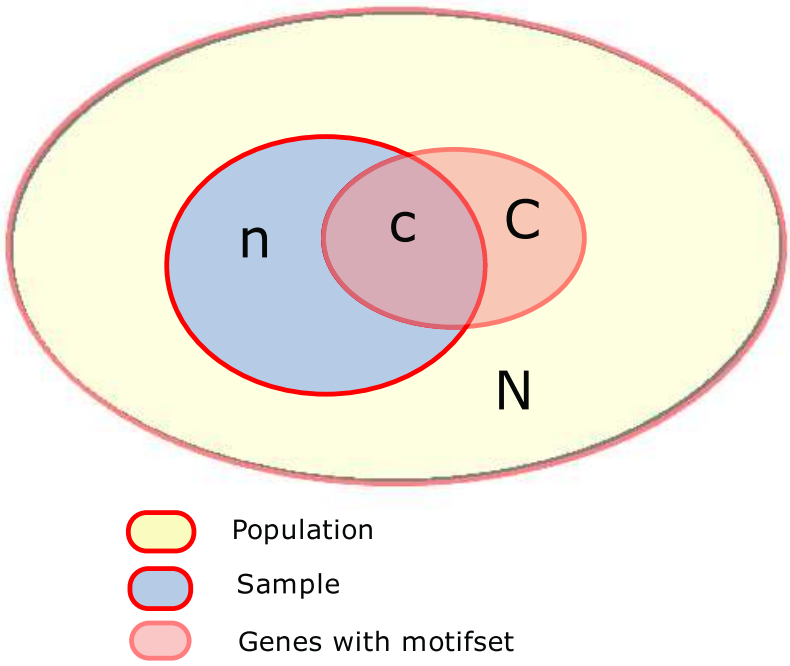


Figure 6.1: The Hypergeometric model

finding more than c objects with a property, from a randomly selected sample of size n is the tail probability of the hypergeometric distribution as calculated by equation 6.8.

$$P[X > c] = \mathcal{H}_{tail}(N, \mathcal{C}_M, n, c) = 1 - \sum_{i=1}^{c} \left(\frac{\binom{n}{c} * \binom{N-n}{\mathcal{C}_M - c}}{\binom{N}{\mathcal{C}_M}} \right) \tag{6.8}$$

6.3 The χ^2 Test of Independence

The chi-square test of independence is employed when a single sample is categorized on two dimensions or variables [Sheskin, 2000]. It is assumed that the sample is randomly selected from the population that it represents. The first and second dimensions consist of r and k categories respectively, that are represented by r rows and k columns of a contingency table, respectively. The test evaluates the general hypothesis that the two variables are independent of each other, or there is zero correlation between them. The chi-square test of independence makes the following assumptions:

1. The sum of the rows and columns are unknown prior to sampling.

Table 6.2: A 2×2 χ^2 contingency table

	C_1	C_2	Row Sums
R_1	$O_{11}(E_{11})$	$O_{12}(E_{12})$	T_{R1}
R_1	$O_{21}(E_{21})$	$O_{22}(E_{22})$	T_{R2}
	T_{C1}	T_{C2}	N

2. Categorical data from mutually exclusive categories are employed.

3. Each subject or object can be represented only once in the data.

4. The expected frequency in each cell is greater than or equal to 5.

The chi-square distribution approximates the exact sampling distribution for the contingency table [Sheskin, 2000]. However, this is true only for large sample sizes. When sample sizes are small, the hypergeometric distribution (Section 6.2) gives an exact probability and should be used instead of the chi-square distribution.

Table 6.2 illustrates the construction of a 2×2 contingency table. O_{ij} represents the observed value in cell (i, j). The expected value in each cell, E_{ij} is calculated as in equation 6.9:

$$E_{ij} = \frac{T_{Ri} \times T_{Cj}}{N} \tag{6.9}$$

The final chi-square score is calculated as in equation 6.10.

$$\chi^2 = \sum\nolimits_{i=1,2,j=1,2} \frac{(E_{i,j} - O_{i,j})^2}{E_{i,j}} \tag{6.10}$$

The obtained value is is evaluated with the table for the chi-square distribution. There are degrees of freedom $df = (r-1)(k-1)$. Table of critical values for χ^2, derived from the right tail of the chi-square distribution are used to reject the null hypothesis only if the value of chi-square is greater than or equal to the tabled critical value at the pre-specified level of significance.

Chapter 7

XcisClique Process Flow

This chapter describes the XcisClique architecture and process flow with a step-by-step account of the methods employed. Section 7.1 explains the processing and correlation of gene expression data in detail. The algorithm used by XcisClique and the processing at each stage are explained by Sections 7.2 to 7.5.3. The probabilistic methods used in the processing have been described in the previous chapter.

Inputs to XcisClique are of three kinds: annotated genome data, regulatory element data, and gene expression data 5. Annotated genome data is curated from NCBI using the scripts `createGeneDB.pl` (Appendix C.1), `createProteinDB.pl` (Appendix C.2), and `ExtractATUpstream.pl` (Appendix C.3). Figure 7 illustrates the preprocessing operations in XcisClique. The perl scripts `FindMotifs.pl` (Appendix D.4) finds genomewide occurrences of *cis*-elements in *Arabidopsis thaliana* and these are fed to the POPS database. This process is incremental for new *cis*-elements. The expression data processing block consists of the perl scripts `MakeExpressionVectors.pl` (Appendix E.1), `CorrelateRepVectors.pl` (Appendix E.2), and `MakeAverageVectors.pl` (Appendix E.3). `MakeExpressionVectors.pl` is used to create log-ratioed expression vectors of genomewide genes from raw intensity data from the Nottingham database. `CorrelateRepVectors.pl` is used to test the goodness of replicate data. `MakeAverageVectors.pl` is used to average gene expression vectors for both replicates of a gene to produce one gene expression vector per gene.

Figure 7 illustrates the general flow of information between the user, the XcisClique package,

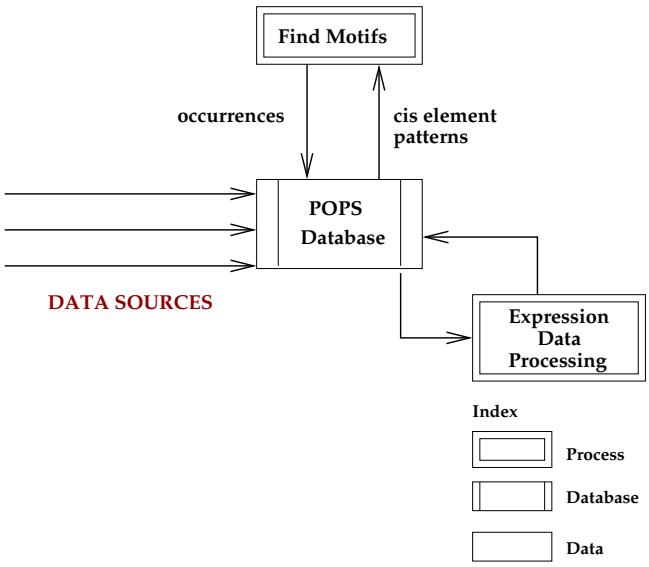


Figure 7.1: XcisClique preprocessing.

and the POPS database during an analysis. Specifications for an analysis include a list of genes, a list of *cis*-elements, a list of treatments, and an FDR. These can be supplied by the users by using data from the POPS database or by creating their own specifications. XcisClique feeds the specifications of a new analysis into the database and results at each stage are also fed into the POPS database. During an analysis, XcisClique uses distributions of the SAV statistics stored in the POPS database. At the end of an analysis, arrangements of motifs on promoters can be visualized using the tool MotifSee, which is available as a web interface.

The XcisClique block in Figure 7 is expanded in Figure 7.3. The `SAV Distributions` block is explained as follows. For a given treatment set, genomewide gene expression vectors are extracted using the perl script `SelectTreatmentVectors.pl` (Appendix E.5). The SAV statistic is sampled using random sets of genes from the genome ranging in size from 2 until the size of the geneset supplied. This is done using the Matlab scripts `CallSimulate.m` (Appendix E.10) and `simulate.m` (Appendix E.11). A distribution for the SAV statistic for the given treatment set is constructed using the perl script `getSDistribution.pl` (Appendix E.12). The `Pairwise Correlations for Geneset` block calculates pairwise Spearman correlations between genes in the geneset using Matlab script `GeneCorrelateInter.m` (Appendix E.7). Promoter sequences of the specified length are ex-

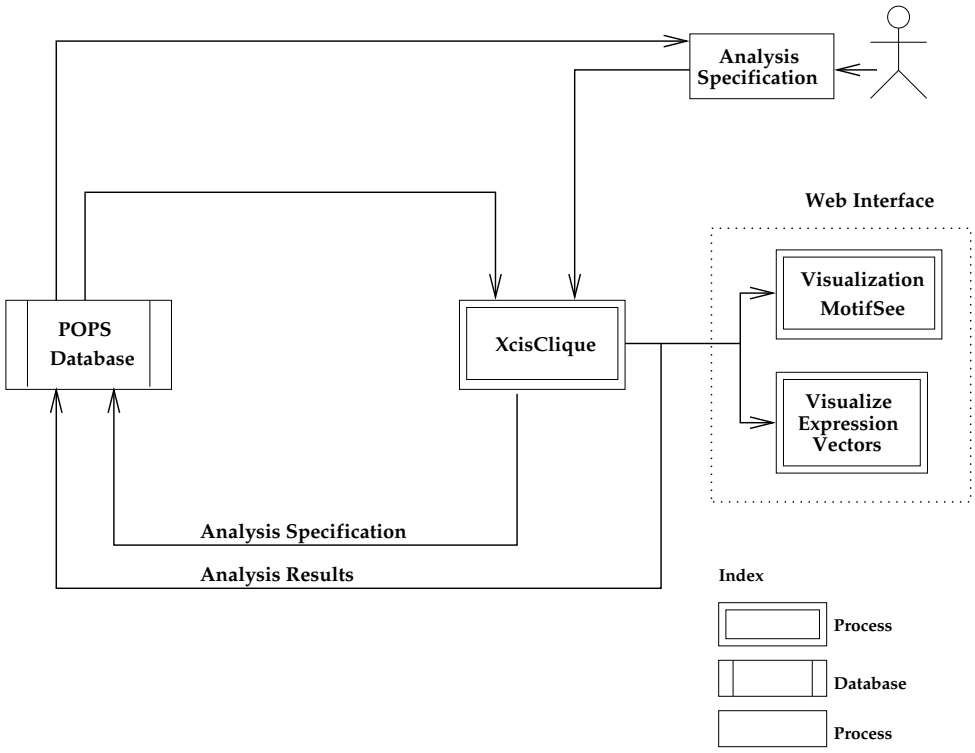


Figure 7.2: XcisClique analysis dataflow.

tracted from the database using the Perl scripts `GetPromoter.pl`, `SelectSequences.pl`, and `XBases` (Appendix D.1, D.2, and D.3). Matches of patterns in these promoters are then located using the Perl script `FindMotifs.pl` (Appendix D.4). Third, the binary matrix for input to the Apriori algorithm is constructed using the Perl scripts `MakeAprMatrix` (Appendix D.7) and the Apriori algorithm is run on this matrix to produce itemsets. Fourth, p-values for over-representation of motifsets are calculated assuming a hypergeometric distribution using the Perl script `Hypergeometric.pl` (Appendix D.8). Fifth, each itemset is evaluated with respect to gene expression data using one of the perl scripts in Appendix D.9. The two p-values for each itemset are combined and the itemsets are ranked.

Figures 7.4 and 7.5 illustrate the flow of information in the XcisClique web interface (`http://bioinformatics.cs.vt.edu/XcisClique`). The web interface allows mining and visualization of existing analyses in the system as well as running new analyses. Figure 7.4 describes the

process flow for the former. Existing analyses in the POPS database are dynamically populated on the website and the user can view details of any of these analyses. Itemsets and their p-values are retrieved from the database. Additionally the user can view motif arrangements on promoters using the visualization tool MotifSee. MotifSee is a PHP based viewer integrated with the XcisClique web interface. The web interface for XcisClique also generates plots for expression vectors of genes in an itemset and these can be visualized.

When a user runs a new analysis on the web interface of XcisClique, he or she has the choice of running the χ^2 test of independence or generating itemsets. Expression data processing is not integrated into new analyses because of its potentially time consuming nature. If an analysis uses a set of *cis*-elements not already present in the POPS database, a request for the analysis is generated and e-mailed to the website administrator. Requests for expression data analysis can also be made.

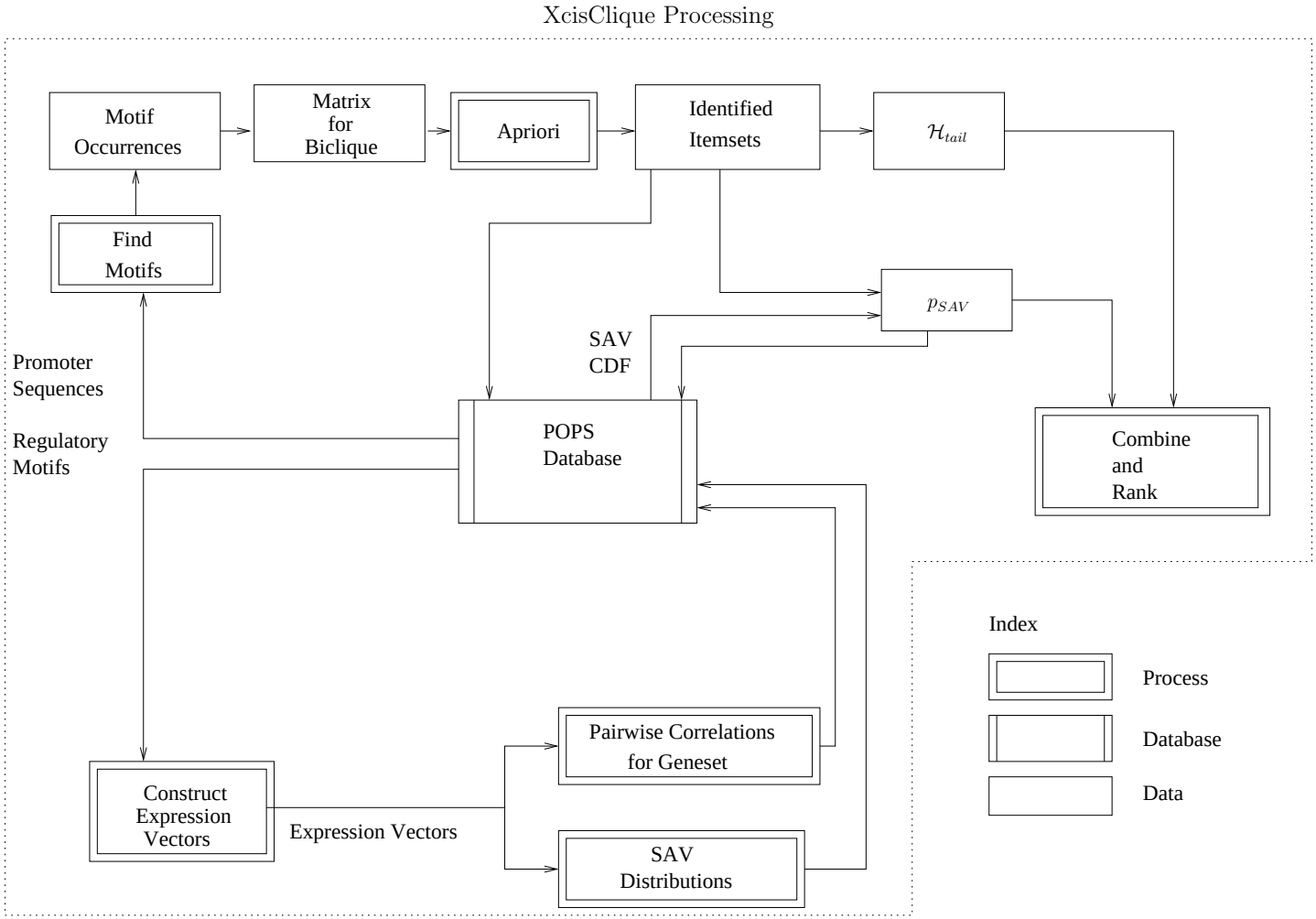


Figure 7.3: XcisClique process pipeline.

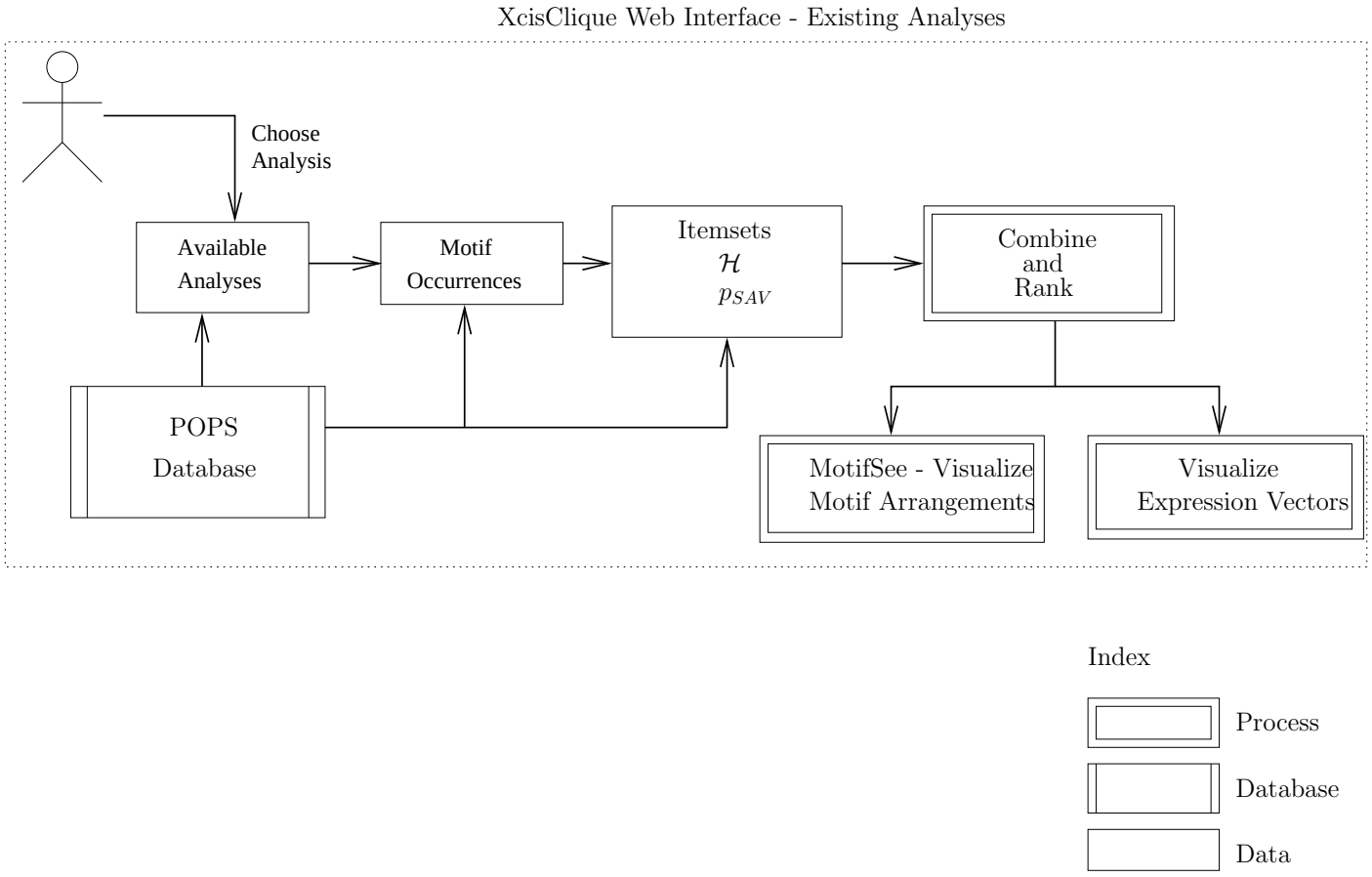


Figure 7.4: XcisClique web interface process pipeline for existing analyses.

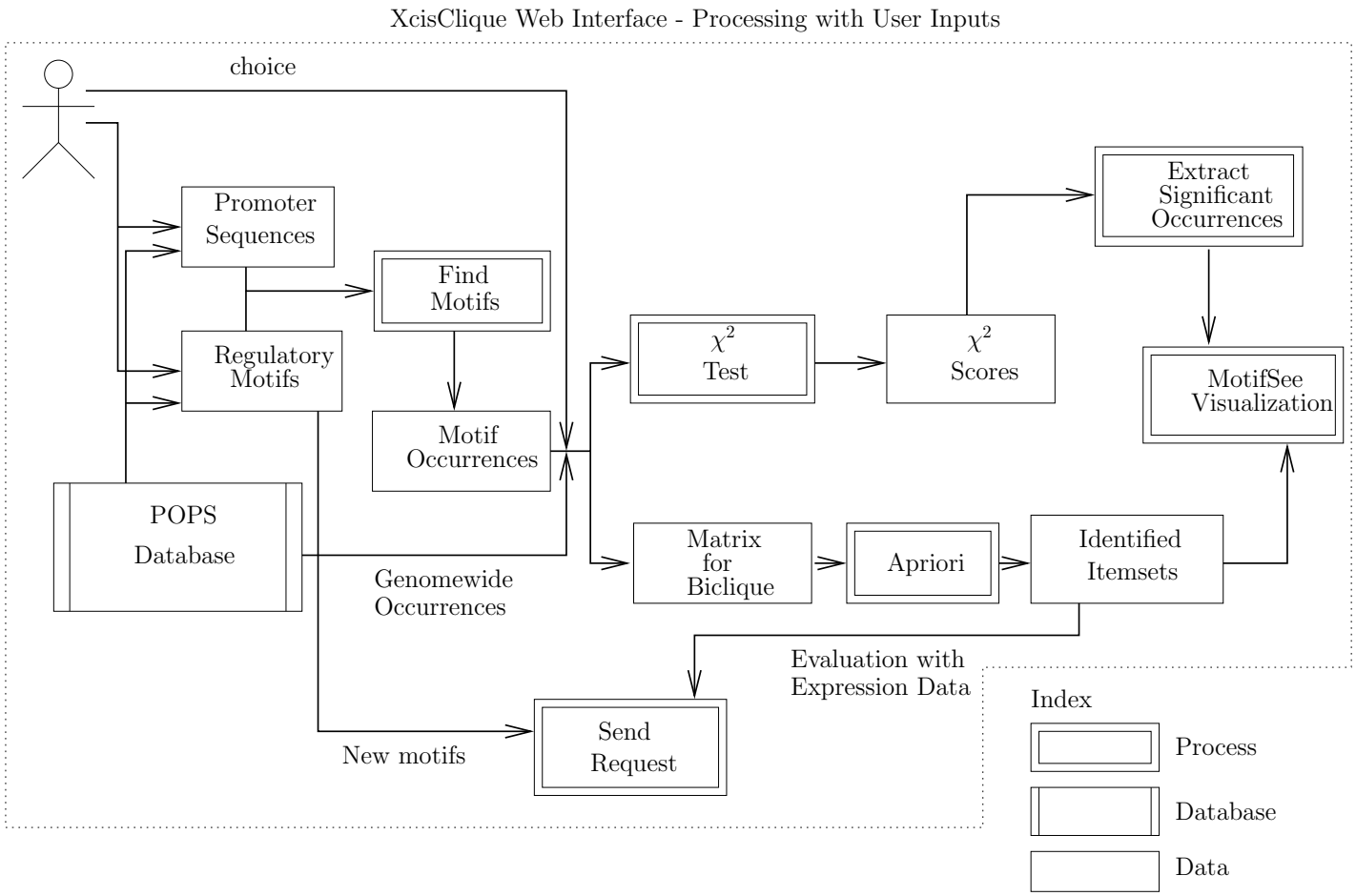


Figure 7.5: XcisClique web interface process pipeline for user-specified analyses.

Table 7.1: Spearman correlation values for biological replicate expression vectors; Roots(R), Shoots(S)

	0.5h		**1.0h**		**3.0h**		**6.0h**		**12.0h**	
Treatment	**R**	**S**	**R**	**S**	**R**	**S**	**R**	**S**	**R**	**S**
Control	.982	.980	.988	.976	.990	.979	.983	.982	.989	.986
Cold	.981	.982	.984	.961	.965	.982	.984	.980	.985	.984
Heat	.985	.983	.988	.974	.984	.979	.989	.986	.987	.983
Drought	.988	.963	.985	.965	.985	.987	.982	.984	.985	.984
Osmotic	.984	.973	.981	.978	.982	.981	.981	.979	.982	.978
Oxidative	.984	.981	.989	.985	.979	.982	.987	.987	.989	.983
Salt	.984	.980	.987	.978	.986	.979	.981	.980	.926	.972
UVB	.990	.977	.992	.979	.992	.986	.991	.987	.987	.988
Genotoxic	.989	.978	.990	.984	.988	.984	.984	.983	.986	.981
Wounding	.987	.983	.988	.976	.989	.984	.980	.979	.986	.981

7.1 Modeling Gene Expression Data

As described in Section 5.2, gene expression data retrieved from NASC were retrieved from the **Nottingham** database on `expresso.cs.vt.edu` on slides corresponding to 9 abiotic stresses, and stored in the POPS database. The slide IDs and the postgreSQL query are available in Appendix B. The expression data thus obtained are processed using the Perl script `MakeExpressionVectors.pl` (Appendix E.1) to extract tissue-specific time series data vectors. A test for goodness of biological replicates was done by computing Spearman (ρ) and Pearson (r) correlation coefficients between replicate vectors for all genes for each $\langle$ `treatment`, `timepoint` $\rangle$ tuple using the Perl program `CorrelateRepVectors.pl` (Appendix E.2). The results of this test are documented in Table 7.1.

Pearson correlations calculated in the same manner have comparable values. Average ρ values for roots and shoots were .9846 and .9806 respectively, and average r values for the same were .9826 and .9908 respectively. According to the critical values tables for ρ and r [Sheskin, 2000], for the correlation of 50 values, as we have above, the critical value of r for a .01 level of significance is .354 and that of ρ is .363. Hence, we conclude that the replicates are highly correlated. Gene expression

vectors were constructed by taking the average of intensity values for both replicates of a gene, using the Perl program `MakeAverageVectors.pl` (Appendix E.3). Root and shoot samples were treated as different biological entities and the genes from these tissues were considered separately.

The level of gene expression is measured by the intensity of the spot corresponding to the probe for a gene on a microarray. There are 9 treatments and 5 time points per treatment. We use $v_{i,j}$ to define the expression of a gene, where j represents the time point and $1 \leq j \leq 5$, and $i \in \{t_1, t_2, \ldots, t_9, c\}$. The type of treatment is specified using i; $i = c$ indicates a control sample whereas $i = t_k,\ 1 \leq k \leq 9$ represents one of the 9 treatments.

Let $\tau = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ represent the set of all 9 treatments. Given a gene g, $v^g_{t_k} = (v_{t_k,1}, v_{t_k,2}, v_{t_k,3}, v_{t_k,4}, v_{t_k,5})$ represents the vector of expression levels of g at different time points under treatment t_k, $1 \leq k \leq 9$. We define the vector concatenation operator $\prod$ such that $\prod_{i=1,2} u_i = [u_1, u_2]$, where u_i are vectors of dimension ≥ 1. The complete expression vector for a gene g over τ is

$$V^\tau_g = \big[v^g_c, \prod\nolimits_{i \leq k \leq 9} v^g_{t_k}\big]. \tag{7.1}$$

The expression vector for a gene g over a selected set of treatments $T \subseteq \tau$ is

$$V^T_g = \big[v^g_c, \prod\nolimits_{t_k \in T} v^g_{t_k}\big]. \tag{7.2}$$

XcisClique offers the flexibility of working with raw intensity vectors or vectors with ratios of treated expression levels to control expression levels. For a time point x, the ratio of the treated expression level to the control expression level for a gene g and any treatment t_k is

$$\nu^g_{t_k,x} = \frac{v^g_{t_k,x}}{v^g_{c,x}}. \tag{7.3}$$

For a treatment t_k, the ratioed expression vector of gene g is

$$\nu^g_{t_k} = (\nu_{t_k,1}, \nu_{t_k,2}, \nu_{t_k,3}, \nu_{t_k,4}, \nu_{t_k,5}) \tag{7.4}$$

From Equations 7.2, 7.3, and 7.4, the ratioed expression vector for gene g over a set of treatments $T \subseteq \tau$ is

$$\Upsilon^T_g = \prod\nolimits_{t_k \in T} \nu^g_{t_k}. \tag{7.5}$$

We observe that $|V^\tau_g| = 50$ whereas $|\Upsilon^\tau_g| = 45$. The relative distribution of rank correlation coefficients (Spearman and Kendall) among genes are not affected by this choice. The complete gene expression vector for a gene consists of all 9 treatments with 5 time-points per treatment. The

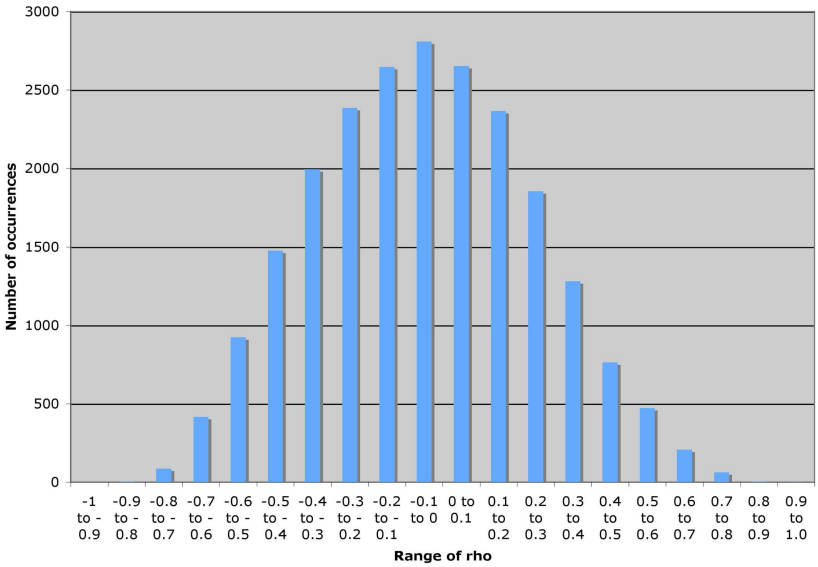


Figure 7.6: Distribution of the ρ value for correlations of the rd29a gene expression vector with all genes of the *Arabidopsis thaliana* genome.

degree to which a pair of genes are co-regulated across a set of treatments can be measured by the correlation between their expression vectors across that set of treatments. The size of the ratioed expression vectors being analyzed can, therefore, range between 5 and 45.

In gene expression correlation in XcisClique, Spearman's correlation coefficient is calculated between ratioed expression vectors of genes. The distribution of ρ-values for the correlation of a gene with all other genes of the genome is approximately normal as indicated by Figure 7.6. Given a pair of genes $g1$ and $g2$, a set of treatments T, and the Spearman correlation coefficient $\rho(g1, g2)$ between Υ^T_{g1} and Υ^T_{g2}, the p-value of correlation can be defined as the probability that Υ^T_{g1} is correlated with Υ^T_{g2} with a correlation coefficient of $\rho(g1, g2)$ by chance alone. This is computed as follows:

$$p(\rho(g1, g2)) = \frac{\rho(g1, g2) - \mu_{(g1,N,\rho)}}{\sigma_{(g1,N,\rho)}} \tag{7.6}$$

where, $\mu_{(g1,N,\rho)}$ and $\sigma_{(g1,N,\rho)}$ are the mean and standard deviations, respectively, of the Spearman correlation coefficients between $g1$ and the all genes of the genome as defined by Equation 2.18.

7.2 XcisClique Inputs

XcisClique consists of programs in Perl, Matlab and C++. The inputs to the system are:

1. A set of AGI numbers corresponding to a geneset of interest to the biologist.
2. A list of regulatory motifs to be searched for. This input is optional and the user can use existing set of regulatory elements in XcisClique.
3. A set of treatments over which expression vectors of genes are to be correlated.
4. Number of upstream bases per promoter.

Upstream regions of length 2000 for the *Arabidopsis thaliana* genome are part of the system and are stored in the database (Section 5.1). Unlike many existing tools, XcisClique does not use the translation start site as the starting point of the gene promoter because there are sometimes many thousand bases between the translation start site and the transcription start site [Berg et al., 2002]. Instead, we have compiled existing and putative TATA box patterns in plants into a regular expression. The regular expression treats the TATA box as an AT-rich sequence containing only As and Ts, beginning with a T and containing at least one A within the first five bases of the sequence. The position of the TATA box is used as a reference point and motifs are found upstream of the TATA box (Appendix A). Regulatory motif sequences can be Perl regular expressions. Examples of input files are available at the website. The script extracts the length of promoters to be analyzed in both foreground (geneset) and background (genome) sequences and finds known regulatory motifs genewise on both strands by Perl pattern matching. Palindromes that result in two occurrences for the same motif when both strands are searched, are recognized and compensated for by the script.

7.3 Analysis of Over-representation of Individual Motifs

χ^2 scores for every motif are calculated by a chi-square test of independence using equation 6.10, where, each of E and O are 2×2-square matrices for observed and expected number of promoters respectively. Columns 1 and 2 indicate presence and absence of the motif respectively, and rows 1 and 2 indicate $\mathcal{F}(A)$ (foreground geneset) and $\mathcal{B}(A)$ (background set, genome\geneset), respectively.

Table 7.3 is an example of a χ^2 contingency table for determining if the `PerfectHSE` motif is overrepresented in the group of 58 genes in *Arabidopsis thaliana* whose expression vectors have

Table 7.2: Contingency table for χ^2 test of independence of the presence of the `PerfectHSE` motif in 58 genes highly correlated with HSP90-1

	`PerfectHSE` present	`PerfectHSE` absent	
Highly correlated with HSP90-1	10(1.0039)	48(47.089)	58
Other	489(497.996)	28,282(28,273.0039)	28,771
	499	28,330	28,829

a Pearson correlation coefficient greater than or equal to 0.70 with the expression vector for the HSP90-1 gene, At5g52640. 15 of the 58 genes in this group have the `PerfectHSE` motif in their upstream regions. The expected motif counts are present in parentheses alongside the observed motif counts. The chi-square score is 80.6358, which at 1 degree of freedom has a probability value less than 0.005. Hence the rows and columns are not independent and a gene being highly correlated to the HSP90-1 gene is not independent of the presence of the `PerfectHSE` motif.

7.4 Identification of Itemsets

Combinations of *cis*-elements that are significantly over-represented in a geneset are identified using the Apriori algorithm. The presence of *cis*-elements in genes is represented by a binary matrix whose rows represent genes and columns represent *cis*-elements. A 1 in cell (i, j) indicates the presence of motif j in gene i. The apriori algorithm finds all maximal groups of 1s in this binary matrix. A set of cell values is called *maximal* when no more rows can be added without removing columns and vice versa. Each motifset-geneset combination output by the algorithm is called an *itemset*. The k^{th} itemset $I_k = \langle G_k, M_k \rangle$ is defined as a biclique with a set of $|M_k|$ motifs, M_k in one clique and a set of $|G_k|$ genes, G_k in the other. Edges connect members of one clique with all members of the other and are representative of the presence of every motif in M_k in every gene in G_k. Table 7.3 illustrates the working of this algorithm with respect to genes and motifs.

The problem of finding maximal itemsets is NP-complete and the implementation in XcisClique is suboptimal. Figure 7.7 illustrates the concept of an itemset of genes and patterns. An itemset does not imply any particular arrangement of patterns. It only indicates the presence of a set of patterns in a set of genes.

Table 7.3: Binary matrix representing a biclique. $G1$ through $G8$ are hypothetical genes and $M1$ through $M5$ are hypothetical motifs. A 1 in cell (i, j) indicates the presence of motif j in the promoter of gene i. A 0 indicates otherwise.

	M1	**M2**	**M3**	**M4**	**M5**
G1	1	1	0	0	0
G2	1	1	1	1	1
G3	0	0	0	0	0
G4	0	0	1	1	1
G5	0	0	0	0	0
G6	0	0	1	1	1
G7	0	0	0	0	0
G8	1	1	1	1	1

Itemsets
⟨M1, M2, G1, G2, G8⟩
⟨M3, M4, M5, G2, G4, G6, G8⟩
⟨M1, M2, M3, M4, M5, G2, G8⟩

7.4.1 Identification of Significant Itemsets

The distribution of a motif combination M in genes across the genome is treated as a hypergeometric distribution. A p-value is generated for each itemset by calculating the tail probability corresponding to the presence of more than c gene promoters with M from n promoters drawn from the genome set of N promoters having C promoters with M, as in equation 6.8.

Itemsets from the output of the Apriori algorithm are subjected to *False Discovery Rate (FDR)* filter [Storey and Tibshirani, 2003]. In multiple hypothesis testing, FDR of a set of predictions is the expected percentage of false predictions in the set of predictions. So, if 100 itemsets are 'discovered' with an FDR of 0.05, then 95 of them can be expected to be correct. The quantity $1 - FDR$ is called the 'confidence' of predictions. Suppose, K itemsets are discovered by apriori and assigned p-values by the hypergeometric distribution tail probability. An FDR correction works by ranking

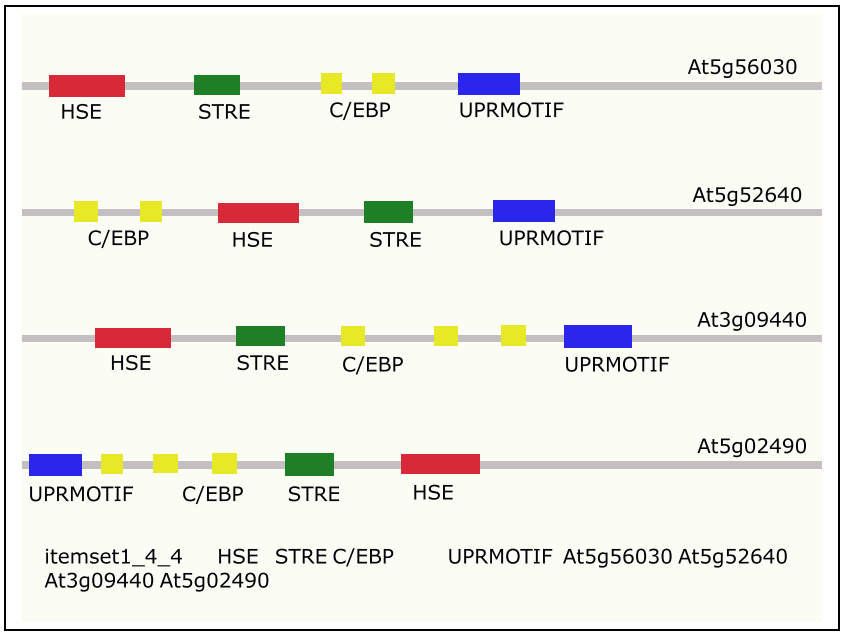


Figure 7.7: Illustration of an itemset

all itemsets in increasing order of their p-values and then marking the i^{th} itemset as a discovery if,

$$p_i \leq \frac{i * FDR}{K} \tag{7.7}$$

and

$$p_j \leq \frac{j * FDR}{K} \tag{7.8}$$

for all $j < i$.

Unlike the Bonferroni correction, which controls the chances of any false positives, FDR controls the expected proportion of false positives and is less stringent than the Bonferroni correction. The default FDR in XcisClique is 5%. This level can be changed in XcisClique inputs to suit user needs.

7.5 Integrating Itemset and Expression Data

Itemsets are evaluated to measure how tightly correlated the genes in the itemset are with respect to gene expression data. XcisClique does this in two different ways as described in Sections 7.5.1 and 7.5.2.

7.5.1 Using p-values of Correlation

This method evaluates the p-value for determining the tightness of correlation of the genes in an itemset by taking the average of p-values of correlations of all *ordered gene pairs* in the itemset as follows:

$$P_{exp} = \frac{\left(\sum_{\substack{1 \leq i \leq |G| \\ 1 \leq j \leq |G| \\ i \neq j}} p(\rho(g_i, g_j)) \right)}{2 * \binom{|G|}{2}} \tag{7.9}$$

where, G is the geneset in the itemset and $p(\rho(g_i, g_j))$ is calculated as in Equation 7.6. The pre-computing required for generating p-values as in Equation 7.9 is outlined by the algorithm in Figure 7.5.1. Here, correlation coefficients of each of the genes with the rest of the genes of the genome are also calculated and the distribution of ρ per gene is stored in the database.

Figure 7.8: Pseudocode for expression data processing to calculate p-values as in Section 7.5.1

INPUTS: A set of genes G, a set of treatments T.

OUTPUT: Distributions (μ, σ) of X for each gene $g_i \in G$, where X is the random variable representing Spearman correlations ρ of g_i with all the genes in the AT genome.

ALGORITHM:

1. Extract treatment vectors Υ_g^T corresponding to T for entire genome.
2. For $i, j = 1 : |G|$, $g_i, g_j \in G$,
   (1) Calculate $\mu_{(g_i,N,\rho)}$.
   (2) Calculate $\sigma_{(g_i,N,\rho)}$.

   End
3. For $i = 1 : |G|$,$j = (i+1) : |G|$
   (1) Calculate $\rho(g_i, g_j)$,
   where g_i and g_j are the i^{th} and j^{th} elements of G respectively
   (2) Assign $\rho(g_j, g_i) = \rho(g_i, g_j)$
   (3) For $\rho(g_i, g_j)$, calculate $P_r[z > \frac{\rho(g_i,g_j)-\mu_{(g_i,N,\rho)}}{\sigma_{(g_i,N,\rho)}}] = 1 - \mathfrak{N}(\rho(g_i, g_j))$
   (4) For $\rho(g_j, g_i)$, calculate $P_r[z > \frac{\rho(g_j,g_i)-\mu_{(g_j,N,\rho)}}{\sigma_{(g_j,N,\rho)}}] = 1 - \mathfrak{N}(\rho(g_j, g_i))$
   (5) Store correlations and p-values in database tables

   End

7.5.2 Sum of Absolute Values of Correlation (SAV): A New Statistic

Consider an analysis $\mathcal{A}$ in XcisClique. Let I be an itemset that consists of geneset G and motifset M. Let $|G|$ be n. Suppose the set of treatments used in $\mathcal{A}$ is T. For any gene $g_i \in G$, the expression vector is $\Upsilon_{g_i}^T$ from Equation 7.5. Let $\rho(g_i, g_j)$ denote the spearman correlation coefficient between $\Upsilon_{g_i}^T$ and $\Upsilon_{g_j}^T$.

We define the SAV (Sum of Absolute Values) statistic S as the sum of the absolute values of Spearman correlation coefficients of all gene pairs in the geneset G of itemset I as follows:

$$S = \sum_{1 \leq i < j \leq n} |\rho(g_i, g_j)|. \tag{7.10}$$

The reasoning behind this statistic is that a high negative correlation is as biologically relevant as a high positive correlation. Binding a class of transcription factors in response to a set of treatments might induce transcription of some genes in the itemset while repressing other genes in the same itemset. The sum of absolute values of ρ is an indication of how tightly (both negatively and positively) correlated the geneset in an itemset is. Figure 7.9 illustrates the probability density function of S and Figure 7.10 illustrates the cumulative distribution function for S, for a geneset of size 6.

For genesets of cardinality less than 16, the mean and standard deviation for S have been calculated by sampling $100,000$ samples of genesets per geneset size. The distribution of the mean of S is an exponential curve whose log-log transformation gives a straight line as illustrated by Figures 7.11 and 7.12. The distribution of the standard deviation of S can also be approximated by a straight line (Figure 7.13).

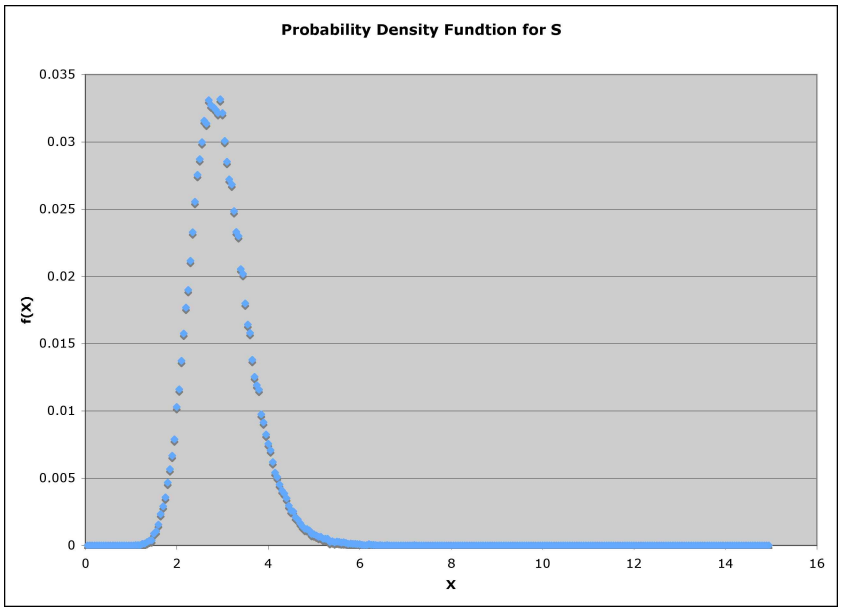


Figure 7.9: Distribution of S for sample size 6: Probability Density Function

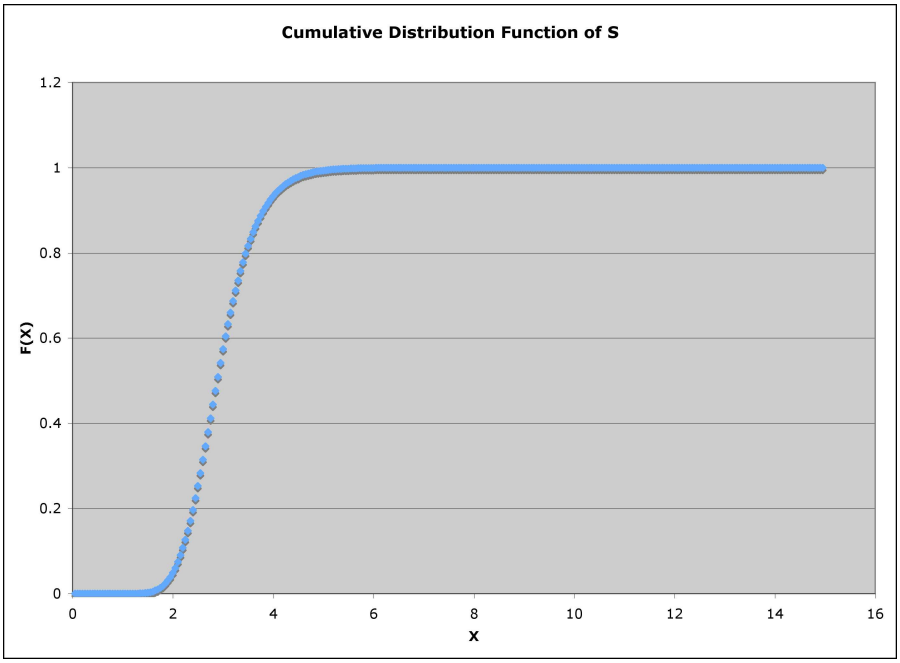


Figure 7.10: Distribution of S for sample size 6: Cumulative Distribution Function

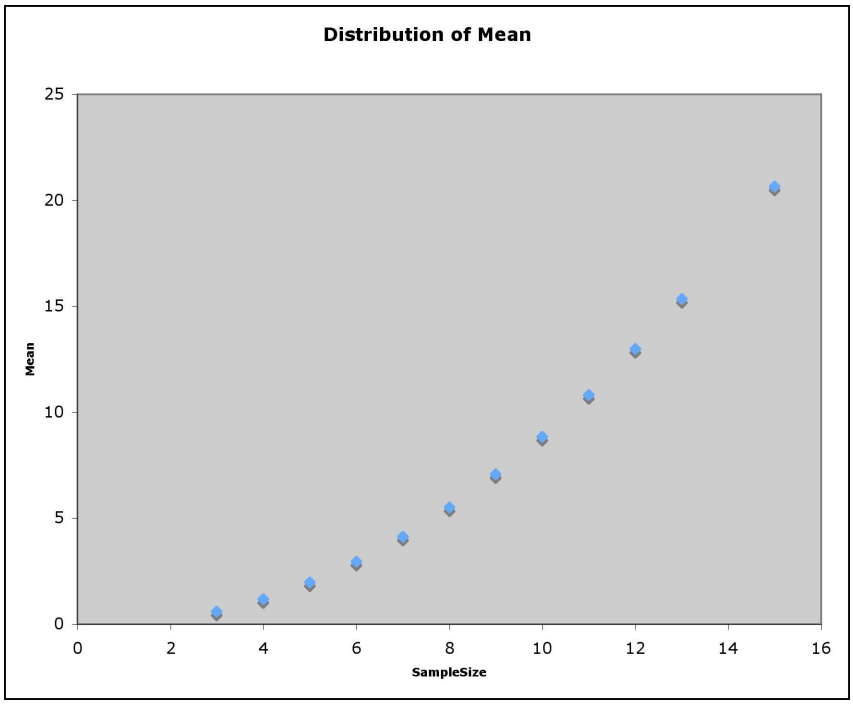


Figure 7.11: Plot of mean vs. sample size for S.

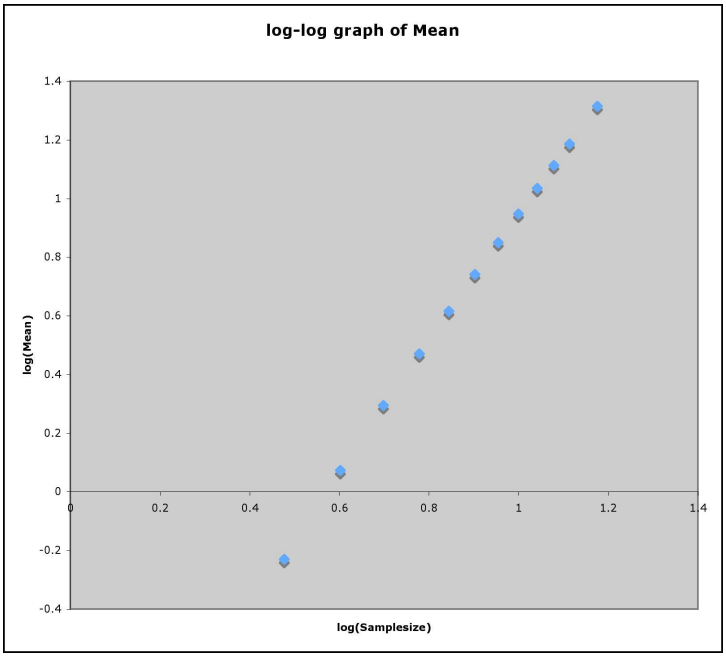


Figure 7.12: log-log plot of mean vs. sample size for S.

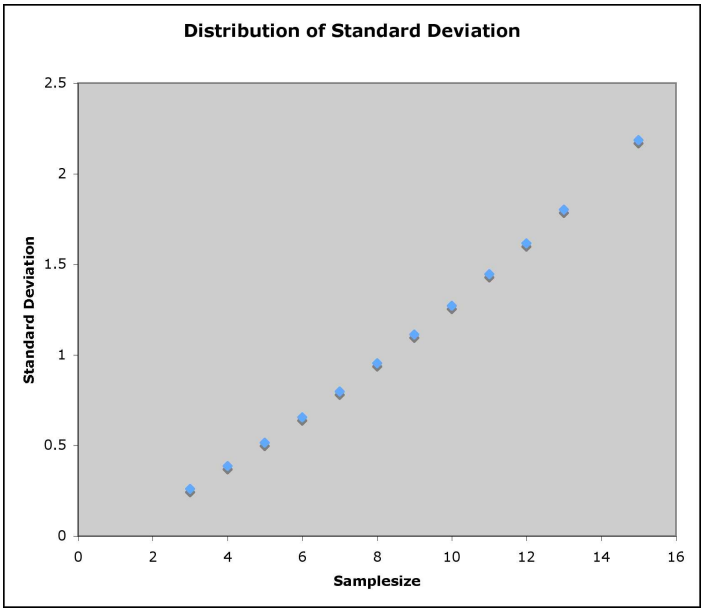


Figure 7.13: Plot of standard deviation vs. sample size for S.

Consider an input set of genes G and an input set of treatments T. First, Υ_g^T is computed for every gene g in G. Next, Spearman correlation coefficients between the expression vectors of all pairs of genes G are computed. To generate the p-value, the SAV statistic is computed on all pairwise Spearman correlations in a geneset. Corresponding to the treatment set T, distributions of the SAV statistic for genesets of sizes $\leq |G|$ are precomputed by random sampling. This is done as follows. First, Υ_g^T is computed for all genes in the *Arabidopsis thaliana* genome. For a given geneset size x, x genes are randomly chosen from the genome and the SAV statistic is computed on this set of random genes. This is repeated $100,000$ times. The MATLAB programs `CallSimulate.m` (Appendix E.10) and `simulate.m` (Appendix E.11) do this. The resulting values of the SAV statistic are used to generate a probability distribution of the SAV statistic for geneset sizes 2 till $|G|$ using Perl script `getSDistribution.pl` (Appendix E.12). This process is repeated for all geneset sizes required. The process is outlined by the algorithm in Figure 7.5.2.

Figure 7.14: Pseudocode for expression data processing to calculate p-values as in Section 7.5.2

INPUTS: A set of genes G, a set of treatments T.
OUTPUT: Distributions of S for genesets of size x, $2 \leq x \leq |G|$, over T.
ALGORITHM:

1. Extract treatment vectors Υ_g^T corresponding to T for entire genome
2. For $x = 1 : |G|$,
   (1) For $i = 1 : 100,000$
   - Randomly sample x genes from the genome
   - Calculate S

   End
   (2) Calculate frequency of S ($0 \leq S \leq \binom{|G|}{2}$) values in intervals of size 0.05
   (3) Calculate $F_S(t)$ for all interval upper limits t
   (4) Store values in database tables

   End

Processing of expression data is time-consuming and so, distributions for S for the set of all treatments τ have been precomputed and stored in the POPS database. The p-value for the SAV statistic s_k of the k^{th} itemset is:

$$p[S \geq s_k] = 1 - F_S(s_k), \tag{7.11}$$

where S is the random variable representing the SAV statistic.

7.5.3 Calculating Final p-values

Consider an analysis A in XcisClique and an itemset I that is a result of the Apriori algorithm. Suppose I consists of a the geneset G and the motifset M. Let N be the number of genes in the *Arabidopsis thaliana* genome and A_G be the number of genes used analysis A. From Section 2.1.2 and 6.2, the number of genes in the genome that have motifset M in their promoters is $\mathcal{C}_M$.

The p-value from expression data analysis of an itemset can be computed in two ways as illustrated in Equations 7.9 and 7.11. In Equation 7.9, p-values of correlation are added. p-values can be added only when the events are mutually exclusive. The correlation of a pair of genes in a geneset might not be exclusive of the correlations of other pairs of genes in that geneset. Hence, the assumption that pairwise correlations are mutually exclusive events cannot be made correctly. In all out future analyses the SAV statistic p-value is used to evaluate an itemset with respect to gene expression data. The final p-value $\mathcal{Q}_I$ for the itemset I is computed as the product of the p-value from the hypergeometric distribution and the p-value from correlation analysis of the itemset. XcisClique analysis follows the expression data processing. In particular, the final p-value for an itemset is:

$$\mathcal{Q}_I \;=\; \mathcal{H}_{tail}(N, \mathcal{C}_M, A_G, |G|) \times p_{exp}, \tag{7.12}$$

where $\mathcal{H}_{tail}(N, \mathcal{C}_M, A_G, |G|)$ is calculated as in Section 6.2 and p_{exp} is calculated from Equation 7.11. Figure 7.5.3 describes the processing in XcisClique.

Figure 7.15: Pseudocode for XcisClique analysis

INPUTS: A set of genes G, a set of patterns P, a set of treatments $T \in \tau$, and the number of upstream bases b, per upstream region.

OUTPUT: A set of relationships

$$\mathcal{R} = \mathcal{R}_i | \mathcal{R}_i = \langle G_i, P_i, \mathcal{Q}_i \rangle$$

ALGORITHM:

1. Extract b bases upstream for every $g_i \in G$, $1 \leq i \leq |G|$
2. For i, j where, $1 \leq i \leq |G|$ and $1 \leq j \leq |P|$
   Find all occurrences of $p_j \in P$ on both strands of $g_i \in G$
   End
3. Create $|G| \times |P|$ binary Apriori Matrix $\mathcal{M}_{gp}$ of genes and motifs
4. Run Apriori algorithm on $\mathcal{M}_{gp}$ to get a set I of itemsets
5. For each Itemset $I_k = \langle G_k, M_k \rangle$, $k = 1 : |I|$
   (1) Calculate $\mathcal{C}_{M_k}$
   (2) Calculate $\mathcal{H}_{tail}(N, |G|, \mathcal{C}_{M_k}, |G_k|)$
   (3) Calculate $S = s_k$
   (4) Calculate $1 - F_S(s_k)$
   (5) Calculate $\mathcal{Q}_k = \mathcal{H}_{tail}(N, |G|, \mathcal{C}_{M_k}, |G_k|) \times (1 - F_S(s_k))$
   (6) Print $\langle G_k, M_k, \mathcal{Q}_k \rangle$
   End

7.6 System Requirements

The following software components are required to install and run XcisClique.

1. Perl 5.8.5 or higher.

2. Perl Modules.

Table 7.4: Perl modules needed by XcisClique

Module Name	**Module Description**	**Availability**
`LWP::Simple`	This Perl module provides a simple, procedural interface to `LWP`, which is the World-Wide Web library for Perl, a set of Perl modules which provides a sample and consistent application programming interface (API) to the World-Wide Web.	CPAN
`Shell`	Perl module to run shell commands transparently within Perl.	CPAN
`DBI`	Perl module for database access. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used.	CPAN
`DBD::Pg`	This is the PostgreSQL database driver for the `DBI` module.	CPAN
`Test::Simple`	Pre-requisite for `DBD::Pg`.	CPAN
`Time::localtime`	Perl module with interfaces to Perl's built-in `localtime()` function.	CPAN
`Math::Matrix`	Perl module with functions for multiplication, inversion, and other common matrix operations.	CPAN

`Statistics::` `Distributions`	Perl module for calculating critical values and upper probabilities of common statistical distributions such as the Normal distribution, the χ^2 distribution, the t distribution, and the F distribution.	CPAN
`PDF`	Perl module with functions for calculating critical values and probabilities of various statistical distributions, such as the Binomial distribution, the Hypergeometric distribution, and the Gaussian distribution.	Packaged with XcisClique.
`Vector`	Perl module for common vector operations and calculation of Pearson and Spearman correlation coefficients between vectors.	Packaged with XcisClique.
`Utilities`	Perl module with common text processing utility functions such as removing white space from a string.	Packaged with XcisClique.

3. PostgreSQL 7.4.7 or higher.

4. MATLAB 7.0.4 with Statistics toolbox.

Chapter 8

Biological Case Studies

8.1 Heat Shock Response

The heat shock response is a conserved and well studied biological response at elevated temperatures. Heat stress causes cellular damages that can lead to cell death, as most enzymes and proteins are denatured or inactivated at high temperatures. Most organisms have developed ways of tolerating and responding to heat stress. In plants, which are sessile, heat stress responses overlap with responses to other stresses such as drought, cold, and oxidative stress [Schoffl et al., 1998]. This chapter starts with a brief overview of different heat shock protein (HSP) families in *Arabidopsis thaliana* (Section 8.1.1) and an introduction to the heat shock response (Section 8.1.2). Section 8.1.3 describes the *cis*-elements involved in heat shock regulation and elucidates the various forms of the heat shock element (HSE). Section 8.1.4 describes putative co-chaperones recognized by analysis of heat shock proteins by XcisClique.

8.1.1 Heat Shock Protein Families

The heat shock response starts with the activation of a heat shock transcription factor, which then induces transcriptional activation of genes that help the cell tolerate stress that can, for instance, avoid protein damage. Heat Shock Proteins (HSPs) are a class of such proteins. These include the HSP90s, HSP70s, HSP40s, HSP60s, HSP100s, and small heat shock proteins.

Compared with other organisms, the HSP families in *Arabidopsis thaliana* are examples of unex-

pectedly large families. In mammals and other eukaryotic organisms, HSP90 forms heterocomplexes with other proteins, called co-chaperones, for the identification and/or activation of target proteins such as the glucocorticoid receptor [Pratt et al., 2001]. These co-chaperones are the heat shock protein organizing complex (Hop), HSP70, HSP40 (J-domain protein), HSP70 interacting protein (HIP), immunophilin, and p23 [Krishna and Gloor, 2001]. Homologs of these co-chaperone genes are present in *Arabidopsis thaliana*, as multi-gene families. The HSP70 family has 18 members [Lin et al., 2001], the J-domain protein family has 89 members [Miernyk, 2001], the Hop family has 3 members [Krishna and Gloor, 2001, Zhang et al., 2003], and the immunophilin family has 52 members [He et al., 2004]. These large families are probably a result of adaptation of plant cells to a range of stresses as HSPs are induced not only under heat stress but also under different environmental stresses, such as drought stress. This adaptation probably involves formation of specific complexes for different stresses. However, this has not been proven for *Arabidopsis* or any other higher plant. Therefore, we will expect similar *cis*-element arrangement among specific HSP90s and their co-chaperones. Using XcisClique, we analyze the possibility of finding candidate co-chaperones for the HSP90 genes.

HSP90s are 90-kDa proteins that are essential molecular chaperones in eukaryotic cells, with key roles in folding and activation of proteins involved in signal transduction and control of the cell cycle [Krishna and Gloor, 2001]. The *Arabidopsis thaliana* HSP90 gene-family consists of 7 HSP90 genes with AtHSP-1 through AtHSP-4 located in the cytoplasm, and AtHSP-5, 6, and 7 located in plastids, mitochondria, and ER, respectively. HSP70s have been shown to exhibit phylogenetic homology in subgroups when grouped based on their cellular compartments [Lin et al., 2001]. Lin et al. [2001] have also indicated that while most of these genes are expressed at a basal level during normal growth, representative genes of each of the sub-groups are expressed at relatively high levels during specific developmental stages and thermal stress. Small HSPs (also called the HSP20 family) are a family of stress proteins that range in size from approximately 16 to 42 kDa. Analysis of the *Arabidopsis thaliana* genome has revealed a total of 13 small HSPs belonging to 6 classes defined on the basis of their intracellular localization and sequences, and 6 ORFs encoding proteins distantly related to the cytosolic class Cl or the plastidial class of small HSPs. These proteins do not show sequence conservation [Scharf et al., 2001]. Therefore small HSPs are different from HSP90s and HSP70s, whose sequence identity is significantly conserved. Also most small HSPs form homo-oligomeric complexes of 200-750 kDa with the size dependent on the specific protein. The HSP40

heat shock protein of 41 kD was first discovered in E. coli. It interacts with the ATP-bound form of DnaK, stimulating ATP hydrolysis to ADP and Pi. This family of proteins is characterized by the presence of conserved domains called DNAJ. It consists of approximately 75 amino-acid residues comprising 4 α-helices. Between the II and the III helix, the HPD tripeptide which is essential for the J-domain activity is found [Miernyk, 2001]. The search of J-domain proteins in the A. thaliana genome, produced an unexpectedly large number of proteins, 89, containing the J-domain. Based on structure comparison with previously reported DnaJ proteins and function prediction, they were classified in 51 families that localized in all the sub-cellular compartments of the cell [Miernyk, 2001].

8.1.2 Heat Shock Regulation

The transcriptional regulation of *H*SP genes is facilitated by one or more Heat Shock Elements (HSEs) located in the promoters of these genes. Comparison of these promoters has suggested that the HSE consists of at least three pentameric units of NGAAN (where N is any nucleotide) arranged as continuous inverted repeats such as -NGAANNTTCNNGAAN-. This pentameric unit is highly conserved in yeast, mold, nematodes, plants, and mammals [Santoro et al., 1998, Grover et al., 2001]. In eukaryotes, presence of multiple HSEs has been observed [Haralampidis et al., 2002, Grover et al., 2001] within 100 bps of the TATA box motif [Schoffl et al., 1998]. Further the HSE could have either an -NGAAN- or an -NTTCN- as the initial pentamer. These functional pentameric units can tolerate an insertion of up to 5 bps in between as long as their orientation is maintained. Heat Shock Response is a highly cooperative phenomenon with multiple HSEs fostering cooperative interaction between HSE trimers. So HSEs can tolerate deviations from the -NGAA- consensus. Mutational analysis has indicated that the G/C bp (G and complementary C) at position one of the unit is more important than the A/T base in the third position [Barros et al., 1992]. Plant HSF1 in Arabidopsis has been shown to bind to these consensus tripartite sequences. It has also been indicated that these consensus HSEs are conserved in the promoters of other stress inducible eukaryotic genes [Bienz and Pelham, 1987, Nover et al., 1987, 2001].

Apart from HSEs a number of other conserved *cis*-elements have been shown to be involved in Heat Shock Regulation. In plants there is evidence for involvement of CCAAT-box elements (called CCAAT-Enhancer Binding Protein), AT-rich sequences, and scaffold attachment regions. Haralampidis et al. [2002] have also indicated the possible role of a few other *cis*-elements such as STREs, AP-1 or MRE binding sites. These *cis*-elements are also known to be involved in stress

Table 8.1: Regulatory elements in heat shock response

Motif Name	**Sequence**
Heat Shock Element (HSE)	GAAnnTTCnnGAA
Stress Response Element (STRE)	CCCCT
Metal Response Element (MRE)	TGCRCNC
Ascorbate Peroxidase Binding Site (AP1)	TGAGTTAG

```
Perfect HSE1     NGAANNTTCNNGAAN
Perfect HSE2     NTTCNNGAANNTTCN
```

Figure 8.1: Perfect HSEs in Perl regular expressions

responses in other organisms and are listed in table 8.1.

In this work we have used concepts outlined above to identify HSEs and other heat shock related *cis*-elements in the promoters of heat shock genes. The significance of these findings have been analyzed with respect to the entire *Arabidopsis thaliana* genome. This is followed by an analysis of the expression levels of these genes with respect to their regulatory regions.

8.1.3 Regular Expressions for HSEs

HSEs have not been defined rigidly in terms of the number of pentanucleotides they might contain, the separation between these pentanucleotides and acceptable mutations in a perfect HSE. A perfect Heat Shock Element is defined as any of the sequences in Fig 8.1. However, perfect HSEs might not be conserved because of mutations or false reads in DNA sequence data. We know that the G/C bp at position one of the unit is more important than A/T base in the third position functionally for the recognition of an HSE by a HSF [Barros et al., 1992]. So allowing mutations in the third positions we get the HSEs as defined in Fig 8.2. Mutations are indicated by capitalized 'N's. Also, according to Santoro et al. [1998], HSEs can tolerate an insertion of about 5 bps between any two pentanucleotides. This brings us to the most general regular expressions that can define an HSE as in Fig 8.3. Table 8.2 lists the Perl regular expressions corresponding to each type of HSE. Since there are no proved results about the suitability of one or more of the above set of HSEs, a test for their significance was done using a Perl script that identifies unique HSEs and other heat shock related

```
Perfect HSE1 Mut1     NGANNNNTCNNGAAN
Perfect HSE2 Mut2     NTTCNNGANNNNTCN
```

Figure 8.2: Perfect HSEs with mutations in Perl regular expressions

```
General HSE1     NGANN{2,7}NTCN{2,7}GAAN
General HSE2     NTTCN{2,7}GANN{2,7}NTCN
```

Figure 8.3: General HSEs in Perl regular expressions

cis-elements in promoters of genes. We hypothesize that HSEs are proximal to the TATA box. The following figure illustrates the distribution of perfect HSEs in the upstream regions of genes of the entire *Arabidopsis thaliana* genome. Clearly we are more likely to find perfect HSEs in the first 500 base pairs from the TATA box as compared to more distal regions of the promoter. So, the script was run on two types of upstream sequences: 500 bp long sequences and whole upstream (2000 bp) sequences of HSP90s,HSP70s, HSP40s, small HSPs, and the entire genome. The results are listed in Table 8.3.

Expected values in the table are calculated based on the number of occurrences in the genome. The precise Perl regular expressions for all HSEs in Table 8.3 are in Table 8.2. In HSP90s, HSP70s, and small HSPs, perfect HSEs are found on the forward strand, whereas, in HSP40s they are found on the reverse strand also. In HSP90s, 70s and sHSPs more mutated HSEs are located on the forward strand and within 500 bases from the TATA box than in HSP40s. General HSEs are over-represented in all families, but general HSEs are also very common in the genome and are present in the promoters of 75% of all *Arabidopsis thaliana* genes. So this pattern might not be of interest in HSP families.

Perl regular expressions used above represent sets of strings. Biologically only the occurrence of some of these strings as *cis*-elements might be relevant. So, the specificity of the regulatory motifs being searched for was increased by using the versions of HSEs in Table 8.4 were also searched.

Table 8.2: Perl regular expressions for HSEs

HSE	**Perl Regular Expression**
Perfect HSE1	GAANNTTCNNGAA
Perfect HSE2	TTCNNGAANNTTC
Mutated HSE1	GANNNNTCNNGAA
Mutated HSE2	TTCNNGANNNNTC
General HSE1	GANN{2,7}NTCN{2,7}GAA
General HSE2	TTCN{2,7}GANN{2,7}NTC

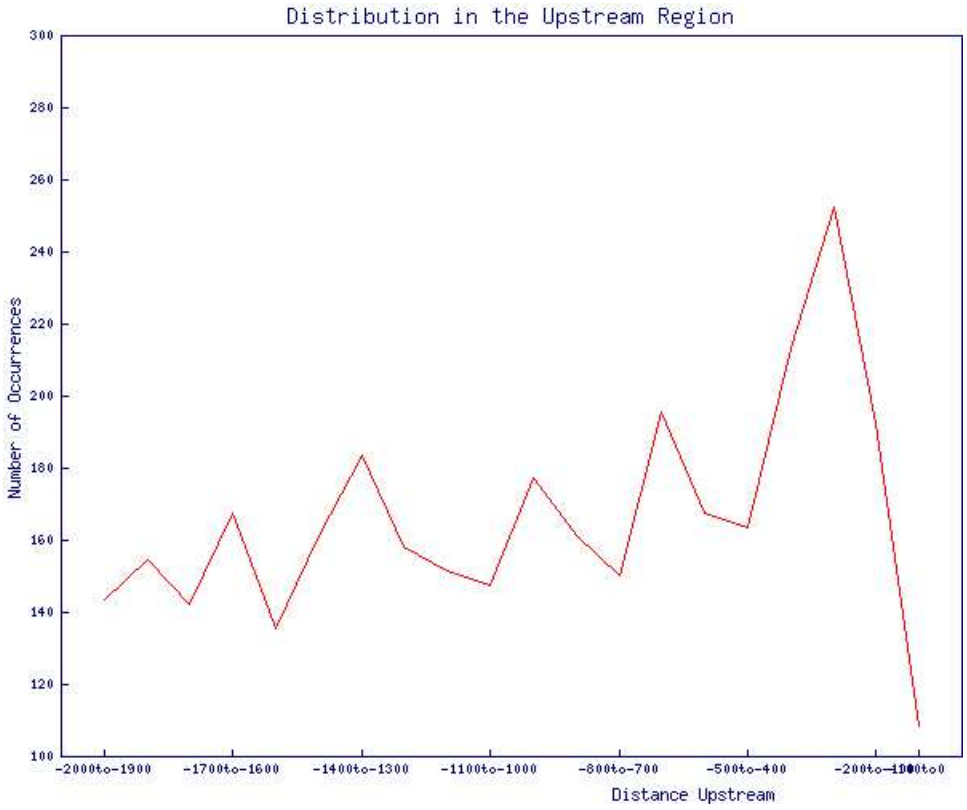


Figure 8.4: Distribution of perfect HSEs in gene upstream regions

Table 8.3: Expected and observed number of occurrences of different HSE patterns in HSP families

		HSP90		**HSP70**		**HSP40**		**small HSP**	
Pattern	**Bases**	**Exp**	**Obs**	**Exp**	**Obs**	**Exp**	**Obs**	**Exp**	**Obs**
Perfect HSE1	500	.027	1	.07	5	.417	1	.074	3
	2000	.118	1	.377	5	.001	1	.324	3
Perfect HSE2	500	.033	0	.086	0	.511	2	.090	0
	2000	.096	0	.307	0	1.479	3	.264	0
Mutated HSE1	500	.187	6	.482	5	2.868	1	.509	7
	2000	.933	6	2.968	9	14.279	10	2.546	7
Mutated HSE2	500	.22	1	.567	3	3.375	5	.599	3
	2000	.931	3	2.959	6	14.237	17	2.546	7
General HSE1	500	6.695	10	13.390	15	79.599	77	14.134	25
	2000	25.511	30	81.902	67	390.127	413	69.275	87
General HSE2	500	5.803	8	11.066	13	68.995	67	12.251	17
	2000	19.891	26	63.227	62	304.18	300	54.013	77

Table 8.4: Different versions of the heat shock element

PerHSE1	(gaa[actg]{2,7}ttc[actg]{2,7}gaa)
LongprePerHSE1	(ttc[actg]{2,7}){0,1}(gaa[actg]{2,7}ttc[actg]2,7){1,2}
	(gaa[actg]{2,7}ttc[actg]{2,7}gaa)
LongpostPerHSE1	(gaa[actg]{2,7}ttc[actg]{2,7}gaa)(ttc[actg]{2,7}gaa[actg]{2,7}){1,2}
	(ttc[actg]{2,7}){0,1}
PerHSE1a	(gaa[actg]{2,7}ttc[actg]{2,7}gaa)([actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)
	[actg]{2,7}gaa){1,2}([actg]{2,7}ttc){0,1}
PerHSE1b	(gaa[actg]{2,7}ttc[actg]{2,7}gaa)([actg]{2,7}ttc[actg]{2,7}
	(gac|gag|gat|aaa|caa|taa)){1,2}([actg]{2,7}ttc){0,1}
PerHSE1c	(gaa[actg]{2,7}ttc[actg]{2,7}gaa)([actg]{2,7}ttc[actg]{2,7}gaa){1,2}
	([actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)){0,1}
PerHSE2	(ttc[actg]{2,7}gaa[actg]{2,7}ttc)
LongprePerHSE2	(gaa[actg]{2,7}){0,1}(ttc[actg]{2,7}gaa[actg]{2,7}){1,2}
	(ttc[actg]{2,7}gaa[actg]{2,7}ttc)
LongpostPerHSE2	(ttc[actg]{2,7}gaa[actg]{2,7}ttc)(gaa[actg]{2,7}ttc[actg]{2,7}){1,2}
	(gaa[actg]{2,7}){0,1}
PerHSE2a	(ttc[actg]{2,7}gaa[actg]{2,7}ttc)([actg]{2,7}(gac|gag|gat|aaa|caa|taa)
	[actg]{2,7}ttc){1,2}([actg]{2,7}gaa){0,1}
PerHSE2b	(ttc[actg]{2,7}gaa[actg]{2,7}ttc)([actg]{2,7}gaa[actg]{2,7}
	(tta|ttt|ttg|atc|gtc|ctc)){1,2}([actg]{2,7}gaa){0,1}
PerHSE2c	(ttc[actg]{2,7}gaa[actg]{2,7}ttc)([actg]{2,7}gaa[actg]{2,7}ttc){1,2}
	([actg]{2,7}(gac|gag|gat|aaa|caa|taa)){0,1}
ImpHSE3	(gaa[actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)[actg]{2,7}gaa)
LongpreImpHSE3	(ttc[actg]{2,7}){0,1}(gaa[actg]{2,7}ttc[actg]{2,7}){1,2}
	(gaa[actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)[actg]{2,7}gaa)
LongpostImpHSE3	(gaa[actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)[actg]{2,7}gaa)
	(ttc[actg]{2,7}gaa[actg]{2,7}){1,2}(ttc[actg]{2,7}){0,1}
ImpHSE3a	(gaa[actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)[actg]{2,7}gaa)([actg]{2,7}
	(tta|ttt|ttg|atc|gtc|ctc)[actg]{2,7}gaa){1,2}([actg]{2,7}ttc){0,1}
ImpHSE3b	(gaa[actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)[actg]{2,7}gaa)([actg]{2,7}
	ttc[actg]{2,7}(gac|gag|gat|aaa|caa|taa)){1,2}([actg]{2,7}ttc){0,1}
ImpHSE3c	(gaa[actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)[actg]{2,7}gaa)([actg]{2,7}
	ttc[actg]{2,7}gaa){1,2}([actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)){0,1}
ImpHSE4	(ttc[actg]{2,7}(gac|gag|gat|aaa|caa|taa)[actg]{2,7}ttc)
LongpreImpHSE4	(gaa[actg]{2,7}){0,1}(ttc[actg]{2,7}gaa[actg]{2,7}){1,2}
	(ttc[actg]{2,7}(gac|gag|gat|aaa|caa|taa)[actg]{2,7}ttc)

LongpostImpHSE4	(ttc[actg]{2,7}(gac|gag|gat|aaa|caa|taa)[actg]{2,7}ttc) (gaa[actg]{2,7}ttc[actg]{2,7}){1,2}(gaa[actg]{2,7}){0,1}
ImpHSE4a	(ttc[actg]{2,7}(gac|gag|gat|aaa|caa|taa)[actg]{2,7}ttc)([actg]{2,7} (gac|gag|gat|aaa|caa|taa)[actg]{2,7}ttc){1,2}([actg]{2,7}gaa){0,1}
ImpHSE4b	(ttc[actg]{2,7}(gac|gag|gat|aaa|caa|taa)[actg]{2,7}ttc)([actg]{2,7} gaa[actg]{2,7}(tta|ttt|ttg|atc|gtc|ctc)){1,2}([actg]{2,7}gaa){0,1}
ImpHSE4c	(ttc[actg]{2,7}(gac|gag|gat|aaa|caa|taa)[actg]{2,7}ttc)([actg]{2,7} gaa[actg]{2,7}ttc){1,2}([actg]{2,7}(gac|gag|gat|aaa|caa|taa)){0,1}

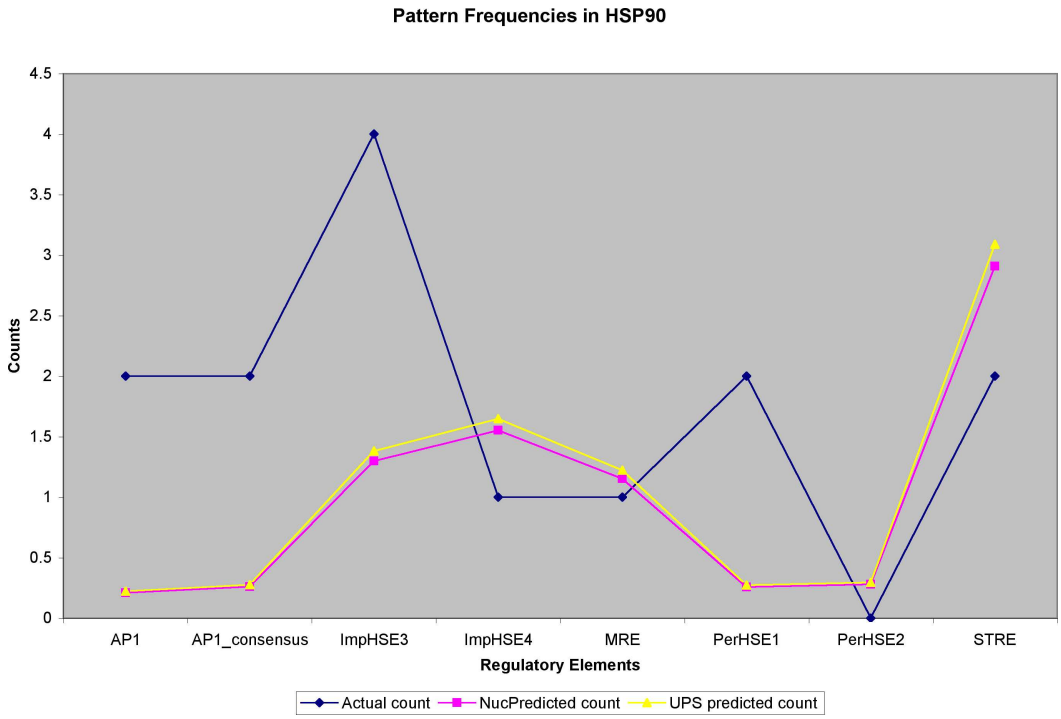


Figure 8.5: Frequency of motifs in HSP90s

Figures 8.5 and 8.6 illustrate the HSE patterns and other regulatory motifs that are overrepresented in HSP90s and 70s, respectively. The series plotted in Figures 8.5 and 8.6 are: the expected counts of motifs taking into account genomewide nucleotide-wise motif counts, the expected counts of motifs taking into account genomewide promoter-wise counts, and the observed counts. Clearly, "PerHSE1" is overrepresented in both the families. "ImpHSE3" is overrepresented in HSP90s. Among other patterns, the "AP1" and "MRE" patterns are overrepresented.

8.1.4 Results of Combinatorial Analysis using XcisClique

Each of the 7 HSP90s were correlated with genes in the rest of the *Arabidopsis thaliana* genome and the top few genes were selected for each gene to be analyzed as a geneset in XcisClique.

Between 60 and 100 top ranked genes were chosen after analysis of correlation for each HSP90s. These genesets were then used with XcisClique to find significant motif combinations. Table 8.5

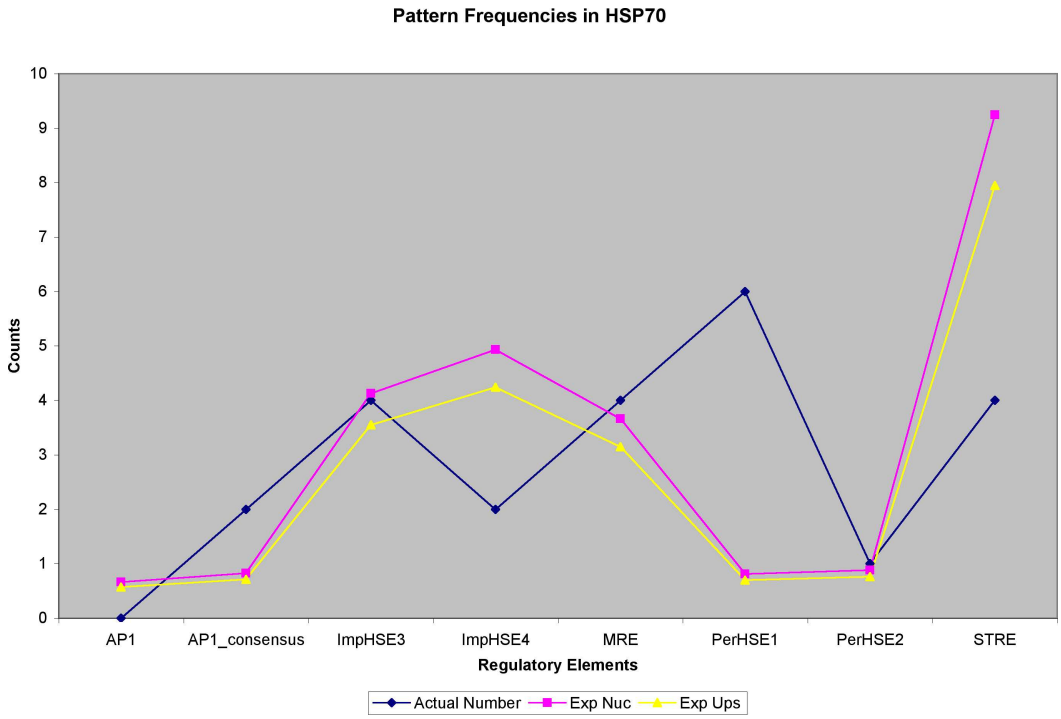


Figure 8.6: Frequency of motifs in HSP70s

summarizes this information. Some significant motif combinations contain each of the HSP90s genes. HSP90-2, HSP90-3, and HSP90-4 share at least 96% similarity at the protein level [Krishna and Gloor, 2001]. At the nucleotide level (coding region) they also show more than 95% similarity. Therefore, only genes that have a specific probe in the Affychip were used in the analysis. For instance, HSP90-4 and HSP90-2 were represented by the same probe on the chip, so only HSP90-4 has been used in the analysis.

Table 8.6 shows significant motifsets found for each group of genes co-regulated with the HSP90s. Previous studies of this family have shown that HSP90-1 was the most induced member by heat shock [Yabe et al., 1994, Haralampidis et al., 2002]. We observe the occurrence of a perfect canonical HSE in the motifsets of the significant itemsets for the gene cluster of HSP90-1. The other cytosolic members (HSP90-2 and HSP90-3) are also induced by heat shock tp levels lesser than HSP90-1 [Milioni and Hatzopoulos, 1997, Prasinos et al., 2005]. Among the significant itemsets for clusters of these HSPs, we find HSEs with 2 mutations and no perfect HSEs. This might explain the

Table 8.5: Analysis of HSP90s with XcisClique

Gene	**At Number**	**Sub-cellular Location**	**Geneset Size**	**Number of Significant Motif Combinations (Spearman ρ cutoff)**	**Number of Significant Motif Combinations with an HSP90**
HSP90-1	At5g52640	Cytoplasm	124(0.6)	77	3
HSP90-2	At5g56030	Cytoplasm	75(0.59)	51	4
HSP90-4	At5g56000	Cytoplasm	48(0.55)	104	7
HSP90-5	At2g04030	Chloroplast	63(0.69)	78	1
HSP90-6	At3g07770	Mitochondrion	79(0.63)	47	6
HSP90-7	At4g24190	Endoplasmic Reticulum	106(0.5)	165	97

reduced response to heat stress. HSPs 6 and 7 are not responsive to heat shock. In the motifsets corresponding to significant motif combinations for clusters corresponding to these genes, no HSEs are found at all.

This analysis also shows putative motifs that might explain the response to other stresses already reported for the HSP90 genes besides heat shock such as water stresses [Kiyosue et al., 1994, Krishna et al., 1995], biotic stresses [Hubert et al., 2003, Takahashi et al., 2003], or embryogenesis [Prasinos et al., 2005] (Table 8.6). Analysis with XcisClique also shows the presence of motifs for responses that have not been reported yet for these genes, such as the circadian cycle related motifs `CCA1ATLHCB1` [Wang et al., 1997] and `EVENINGAT` [Harmer et al., 2000]. These motifs together with the light responsive elements `TBOXATGAPB` [CS et al., 2001] and `HDZIP2ATATHB2` [Ohgishi et al., 2001] are present in the organellar HSP90s (HSP90-6 and HSP90-7), which suggests that these genes might have a role in the folding or activation of proteins that regulate these responses.

Putative co-chaperones

XcisClique analysis helped locate at least one putative co-chaperone for 4 HSP90 genes. These genes

shared high correlations and shared similar motif combinations. Table 8.7 shows the putative co-chaperones found. We were able to find at least two co-chaperones for HSP90-4 and HSP90-7. The co-chaperones computationally assigned in the case of HSP90-1 and HSP90-4 localize in the same cellular compartment. But the same is not true for HSP90-6, where the gene is predicted to localize in the mitochondria [Krishna and Gloor, 2001] and the co-chaperone found is predicted to localize in the chloroplast. There is no experimental data confirming these predictions, so the prediction of the localization might be incorrect.

We were not able to find a co-chaperone for HSP90-5. This might be because the motifs that co-regulate HSP90-5 and its co-chaperones have not been discovered yet, and hence are not in the motif set for XcisClique, or that this HSP90 does not form a complex with co-chaperones.

Table 8.6: Significant motif combinations in HSP90 co-regulated genes

Gene	**Heat**	**Light**	**Circadian Cycle**	**ABA, Drought, Cold**	**GA (embryogenesis)**	**UPR**	**Biotic Stress**	**Organ Development**
HSP90-1	PerfectHSE1[1]			MYB1AT				
HSP90-2	2-mutPerHSE			MYB1AT, Fed-ABRE-like, Fed-AtMyb4, MYB2-CONSENSUSAT, MYCCONSENSUSAT, DPBFCOREDCDC3, DRECRTCOREAT, LTREATLTI78	GAREAT		Fed-WRKY-like	
HSP90-4	2-mutPerHSE			MYB1AT, MYB2-CONSENSUSAT, MYCCONSENSUSAT	GAREAT		Fed-WRKY-like	
HSP90-5		TBOXATGAPB	CCA1AT-LHCB1	DPBFCORE-DCDC3, DRECRTCOREAT				L1BOXATPDF1
HSP90-6	TBOXATGAPB, HDZIP2AT-ATHB2	CCA1AT-LHCB1, EVENINGAT	MYB1AT Fed-AtMyb4	GADOWNAT		Fed-WRKY-like,Fed-TGA1	CARGCW8GAT	
HSP90-7				MYB1AT, Fed-ABRE-like, DPBFCORED-CDC3		UPRMOTIFIIAT		

Table 8.7: Identification of co-Chaperones by XcisClique

Gene Name	**HSP90 Location**	**HSP70**	**Ppiase**	**DNAJ**	**sHSP**	**Other Stress Proteins**	**Non-Stress Proteins**
HSP90-1	Cytoplasm	At3g12580 (Cyt)[2], At5g02490 (Cyt)			At3g46230 (AtHsp17.4-Cl, ND), At1g53540 (AtHsp17.6C-Cl, ND)	At2g26150 (AtHsfA2, Nu), At3g24170 (Glutathione Reductase, Cyt)	
HSP90-4	Cytoplasm	At3g12580 (Cyt)	At3g25230 (ND)			At3g12050 (Activator of HSP90, ND), At4g30490 (AFG1-like ATPase family, Mt)	At3g53550 (Hypothetical Protein, ND)
HSP90-5	Chloro-plast						At5g61170 (40s Ribosomal protein S19, ND), At4g18730 (60s Ribosomal protein, Cyt)
HSP90-6	Mitochondrion	At5g49910 (Chl)					
HSP90-7	Endoplasmic Reticulum	At4g166609 (ER)		At3g62600 (ER)		At1g56340 (Calreticukin, Mt), At1g21750 (Protein disulfide isomerase, ER), At5g61790 (calnexin, Mt)	At2g02810 (UDP-galactose/ UDP-glucose transporter, OM), At1g56330 (GTP-binding protein, ER)
Key	ER: Endoplasmic Reticulum, Cyt: Cytosol, Chl: Chloroplast, Mt: Mitochondria, ND: Not Defined, Nu: Nucleus, OM: Other Membranes						

Other Stress Genes Present in the Itemsets

Putative co-chaperones such as sHSPs for HSP90-1 are present in the significant itemsets. Lee and Vierling (2000) have shown cooperation between sHSPs and HSP70s in folding heat denatured proteins in vitro. These genes appear together with HSP90-1 in an itemset that contains a perfect HSE. HSP90-1 is the most heat inducible HSP90 [Haralampidis et al., 2002] therefore; we could infer that the primary role of this protein is exerted under heat stress together with HSP70s and sHSPs.

Genes that are related to unfolded protein response (UPR), such as protein disulfide isomerase, calreticulin, and calnexin (Table 8.6) appear to be co-regulated with HSP90-7. All these genes share the unfolded protein response motif `UPRMOTIFIIAT` found in genes up-regulated after ER stress induction by tunicamycin [Martinez and Chrispeels, 2003]. But two of the genes grouped with this gene: At1g56340 (calreticulin) and At5g61790 (calnexin) do not have the KDEL ER-targeting peptide. At5g61790 was upregulated after tunicamycin treatment [Martinez and Chrispeels, 2003], which supports the association of this motif combination with unfolded protein response. At1g56340 and At5g61790 are predicted to localize in the mitochondria. Calreticulin genes are known to be chaperones and calcium binding proteins in mammals, regulating many processes [Persson et al., 2003], but in plants the function for many of them is not known. So we can infer that the co-expression of these genes with HSP90-7 might be part of the unfolded protein response signaling pathway, where calcium is also a secondary messenger [Schrode and Kaufman, 2005] or part of acting as chaperones in a subcellular compartment different from the ER. Putative co-chaperones can be verified by y2h (Yeast 2-hybrid) experiments.

More Discussion

Functional redundancy of cytosolic HSP90s is evident as the protein and nucleotide coding region of the HSP90-2, HSP90-3, and HSP90-4 show more than 90% similarity. However the alignment of the 5' UTR and 500bp promoter region does not show that much similarity (less than 50% in both cases). These genes are probably a consequence of gene duplication, but the promoter region has not been conserved. This explains why a point mutation in one HSP90 (HSP90-2) can impair the transduction of the hypersensitive response pathway in Arabidopsis [Hubert et al., 2003]. Therefore, analysis of the expression of these genes should be done to get more information about the role of these proteins. This analysis can be done by real time PCR and designing primers in the regions that are not homologous, such as the 5' UTR.

Organellar HSP90s: Old and New Roles

Cao et al. [2003] have shown that HSP90-5 is induced by light and a mutation in this gene (cr88) inhibits development of the chloroplast. This mutant is also defective in other aspects of photomorphogenesis. The significant motifs found for this gene include the light responsive element (`TBOXGAPB`) and the `L1BOXATPDF1` motif, which are found specifically in the L1 layer expressed protodermal factor1 (PDF1) gene. These motifs can explain the induction by light and the defects in photomorphogenesis by Cao et al. [2003].

Cao et al. [2003] have also shown that HSP90-5 is induced by heat stress. But they have also shown that the presence of this gene in the wild type does not confer higher thermo tolerance as compared with the mutant genotype (cr88) that has the HSP90-5 gene knocked out. The absence of an HSE might be held responsible since we could not find an HSE among motifsets in the significant itemsets for this gene. Therefore it is possible for this gene to have a role in light response. Functional genomic data is not available for HSP90-6 yet. Besides the light and circadian cycle motif, the `CARGCW8GAT` that is the binding site of Agamous-like 15 protein [Tang and Perry, 2003], appears as part of a significant motif combination. Therefore from the significant motifs found for this gene, we could predict that a mutated or over-expressed HSP90-6 genotype could have altered light, circadian cycle responses and flower development. A mutation in HSP90-7 produced and altered flower development phenotype [Ishiguro et al., 2002]. Ishiguro et al. [2002] associated the phenotype with the Clavata3 gene, that regulates flower development, being a target of this HSP90. Therefore a role in flower development for HSP90-6 cannot be discarded.

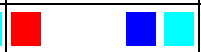
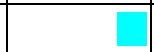
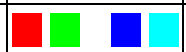
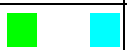
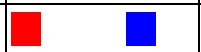
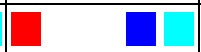
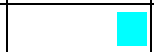
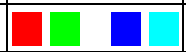
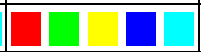
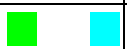
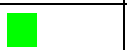
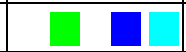
8.2 Cold Long Term Up-regulated Genes in Metabolism

Using XcisClique, we analyzed a set of 11 Arabidopsis genes involved in carbohydrate-metabolism and secondary-metabolism, that are up-regulated long-term by cold stress [Fowler and Thomashow, 2002]. The Apriori algorithm identified 117 itemsets. Treating the distribution of itemsets as hypergeometric along with an FDR of 0.05 gives 81 significant itemsets. On correlation with expression vectors from shoots, 46 itemsets are found significantly correlated with respect to the final p-value. This number for roots is 44. Table 8.8 lists the 5 most significant motif combinations found as part of the itemsets with shoot expression data. These include `CRT` or `DRE` like elements where the inducible transcription factors CBF1, CBF2, and CBF3 bind [Baker et al., 1994, Sakuma et al.,

2002], as well as motifs associated with other abiotic stresses such as water stress (ABRE like motifs), and, unexpectedly, motifs that have been discovered in pathogen or salicylic acid responsive genes, such as WBOXATNPR1 [Yu et al., 2001] and ASF1MOTIFCAMV elements. Among the genes included in the analysis, 5 genes are flavonoid synthesis enzymes. This could explain the presence of pathogen-related motifs, since different flavonoids (i.e. isoflavonoids) are known to be induced after pathogen attack [Camera et al., 2004, Laloi et al., 2004] . Another motif that was not expected to be found was CCA1ATLHCB1, binding site of the Circadian Cycle Associated protein (CCA1), a Myb-related transcription factor [Wang et al., 1997]. Recent studies on cold-response in Arabidopsis have shown that CBF transcription factors are regulated by the circadian cycle, with the highest expression when the plants are transfered to a lower temperature 4 hours after dawn [Fowler et al., 2005]. The CCA1 transcription factor is responsible for the transduction of the phytochrome response [Wang et al., 1997], and probably in combination with other transcription factors induces CBF expression. The genes analyzed in this group are not CBF transcription factors, but two of the genes that contain the CCA1ATLHCB1 motif (Table 8.8) increase their expression within one hour of stress (Figure 8.7) and Fowler and Thomashow [2002], gradually increasing until they reach a maximum after 12 hours. Therefore, the initial response of these genes might be due to CCA1 induction and the peak reached by CBF induction.

All genes that belong to an itemset are highly correlated among themselves with a maximum p-value of 0.0025. p-values of expression data correlation were calculated using Equation 7.9. Itemset 28 is particularly interesting because its tight co-regulation is supported not only by AtGeneExpress data but also by Fowler and Thomashow [2002] and Vogel et al. [2004]'s results. These genes are part of the CBF regulon and show long term up-regulation in shoots and roots after cold stress. They are also induced under normal conditions by CBF over-expression [Vogel et al., 2004]. Figure 8.7 shows the similar expression of these genes under different abiotic stresses in roots and shoots. Within the first 500bp of these genes (Figure 8.8), the Fed-ABRE like motif [Mahalingam et al., 2003] is followed by a Fed-AtMyb4 motif [Mahalingam et al., 2003] within less than 100bp. This order may explain the similar expression of genes in roots and shoots. These genes are part of the CBF regulon, but only the At1g09350 gene has the DRECOREAT motif within 1200bp of the putative promoter region. Vogel et al. [2004] were not able to find putative CBF sites for 20% of the genes that were shown to be cold responsive and induced by CBF overexpression. It is possible that these two genes are part of that set of genes that do not have the CBF binding site within 1200bp and might have it

Table 8.8: Selected significant motif combinations from Cold up-regulated metabolism genes in shoots

	At1g60470	At1g62570	At1g09350	At4g27560	At5g20830	At2g47180	At2g16890	At3g51240
CCA1ATLHCB1								
Fed-AtMyb4								
MYB1LEPR								
RAV1AAT								
WBOXATNPR1								
ZAT12-down								
ABRELATERD1								
Fed-ABRE-like								
C/EBP								
GAREAT								
ASF1MOTIFCAMV								
ACGTABREMOTIFA20OSEM								

Key	:Itemset 8, $p = 8.76e-04$	:Itemset 28, $p = 4.58e-06$	: Itemset 67, $p = 1.01e-04$	: Itemset 87, $p = 8.92e-07$	: Itemset 125, $p = 1.01e-05$

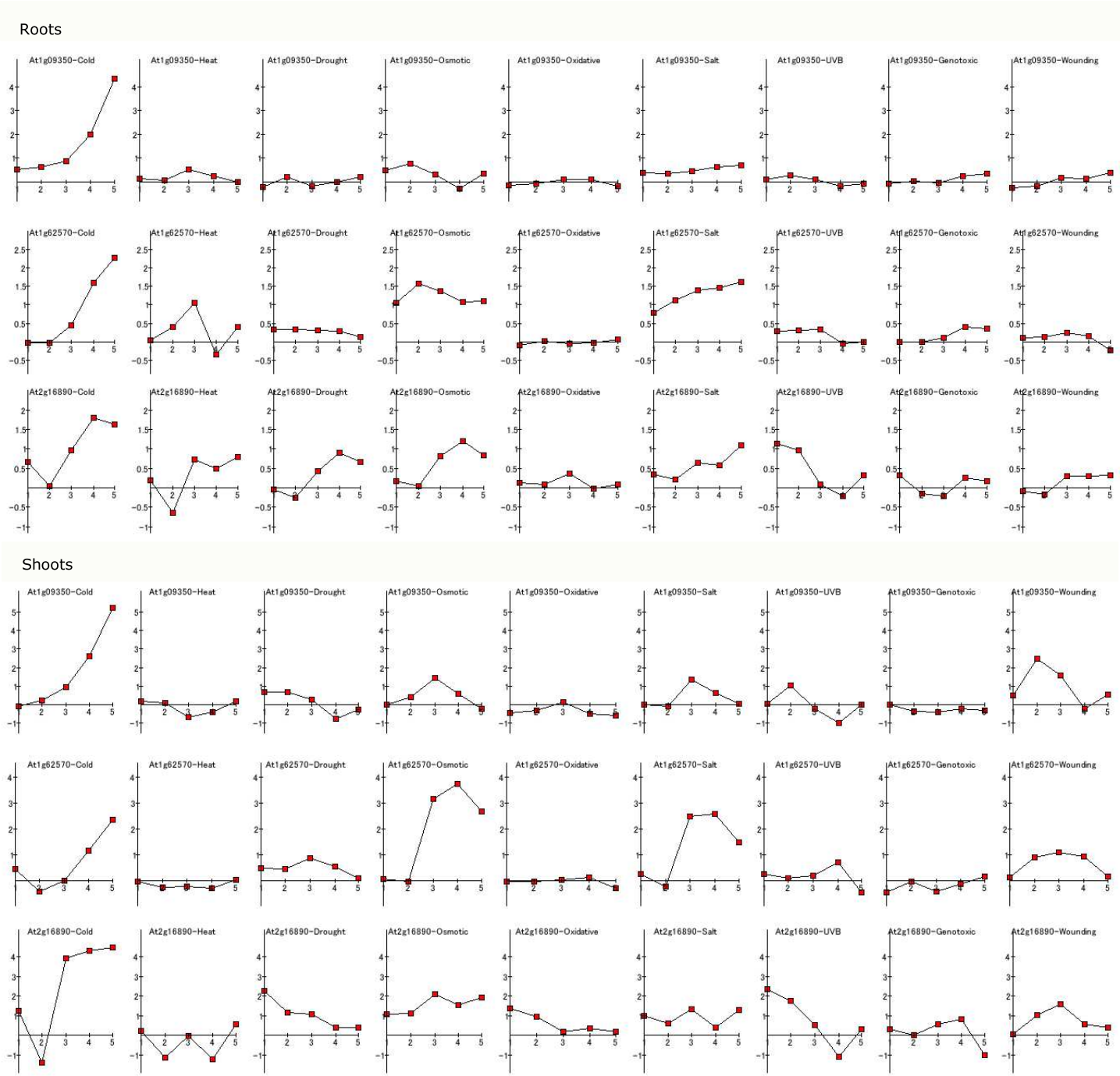


Figure 8.7: Expression vectors for genes in Itemset 28

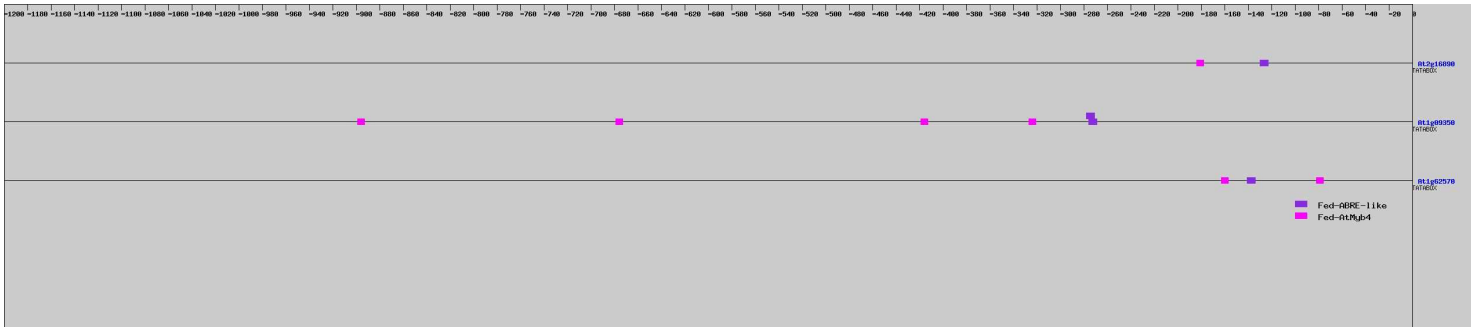


Figure 8.8: Motif arrangements Itemset 28

Figure 8.9: Motif arrangements in Itemset 8

further upstream or before the TATA box. Itemset 8 is also interesting because it also contains 3 genes that are part of the CBF regulon [Vogel et al., 2004] and the motifs contained in this set follow an order: `MYB1LEPR`, `WBOX` and `CCA1TATLHCB1` (Figure 8.9). Two of the genes that belong to this motif (At1g62570 and At1g60470) are putative galactinol synthase genes. These genes are part of the raffinose biosynthesis pathway which accumulates in plants treated by cold and drought [Taji et al., 2002]; raffinose is a sugar that acts as an osmoprotectant under cold and drought.

8.3 Cold Long Term Down-regulated genes in Metabolism

The promoters of 14 genes that have shown down-regulation after cold stress application [Fowler and Thomashow, 2002] were analyzed through XcisClique. The Apriori algorithm identified 169 itemsets, which, after a correction with an FDR of 0.05 resulted in 140 significant itemsets. On correlation with expression vectors among genes, 67 significant itemsets were obtained for shoots and 61 for roots. Table 8.9 shows selected significant motif combinations found for these genes in shoots and roots. p-values of expression data correlation were calculated using Equation 7.9. Vogel et al. [2004] have shown that many of the genes that were down-regulated by cold were also down-regulated by over-expression of the CBF or the ZAT12 transcription factors. They found putative motifs responsible for down-regulation, but none of the genes that we have studied were shown to be down-regulated by over-expression of CBF and ZAT12 transcription factors [Vogel et al., 2004]. This explains why we did not find these motifs in promoters of these genes, and even if they might be present individually, their presence was not associated with a significant motif combination. Genes in itemset 30 show down-regulation under cold stress, but also up-regulation under salt stress (Figure 8.10). This response could be explained by the presence of the combination, `ABRELATERD1`, `MYCATRD22`, and `RAV1AAT`, which are binding sites of transcription factors responsive to ABA, drought and cold [Zhu, 2002], respectively, but have also been found in salt stress responsive genes [Zhu, 2002]. Itemset 12 shows a motif arrangement that includes a mutated version of the heat shock element. The heat shock element binding site is formed by alternate repeats of the pentamer 5'-nCCGn-3'. It allows a mutation in the A/T nucleotides of the pentamer [Santoro et al., 1998]. The HSE motif found in these genes shows a mutation in an A/T in the 1st and the 3rd pentamer of the element. These genes also show down-regulation under heat stress (Figure 8.11). Therefore this HSE motif might be a specific binding site for the class B of heat shock factors, that are negative

Table 8.9: Selected Significant motif combinations from Cold down-regulated metabolism genes in shoots

	At1g55920	At4g15550	At5g14740	At5g24160
ABRELATERD1	■ ■	■	■	■ ■
ASFIMOTIFCAMV	■ ■	■	■	■ ■
C/EBP	■ ■	■	■	■ ■
Fed-SA-induced	■	■		■
IBOX	■ ■	■	■	■ ■
MYCATRD22	■	■		■
RAV1AAT	■ ■	■	■	■ ■
WBOXATNPR1	■	■	■	■ ■
MRE	■		■	■
TELOBOXATEEF1AA1	■		■	■
Key	■ :Itemset 30, $p = 7.17e-05$		■ :Itemset 12, $p = 2.62e-05$	

regulators of transcription [Czarnecka-Verner et al., 2004].

8.4 Senescence genes involved in Stress, Pathogenicity, and Secondary Metabolism

An input set of 17 senescence genes in *Arabidopsis thaliana* were analyzed using XcisClique. These genes are taken from Gepstein et al. [2003], and are involved in stress, pathogenicity, and secondary metabolism. Expression data for 15 of the 17 genes is available. Promoters of length 1200 for the input geneset were scanned for the set of all *Arabidopsis thaliana cis*-elements. Expression data were correlated over a set of 7 treatments (Cold, Heat, Drought, Osmotic, Oxidative, Salt, UVB) in shoots. The complete set of results for this analysis can be viewed at the website. p-values for expression data correlations were calculated using Equations 7.10 and 7.11. Table 8.10 shows a selected set of 3 itemsets that are significant both by sequence and expression data analysis.

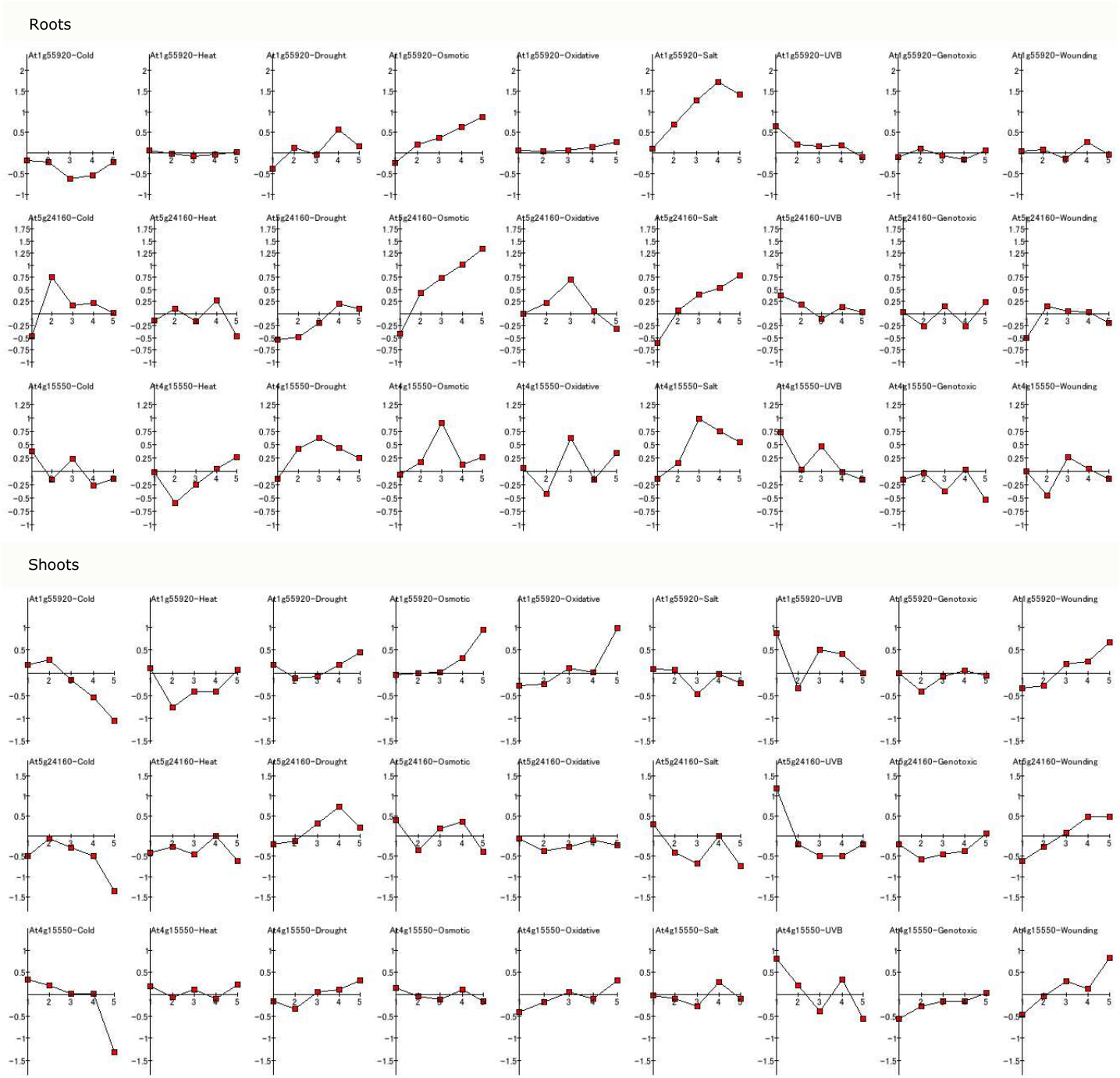


Figure 8.10: Expression vectors for genes in Itemset 30

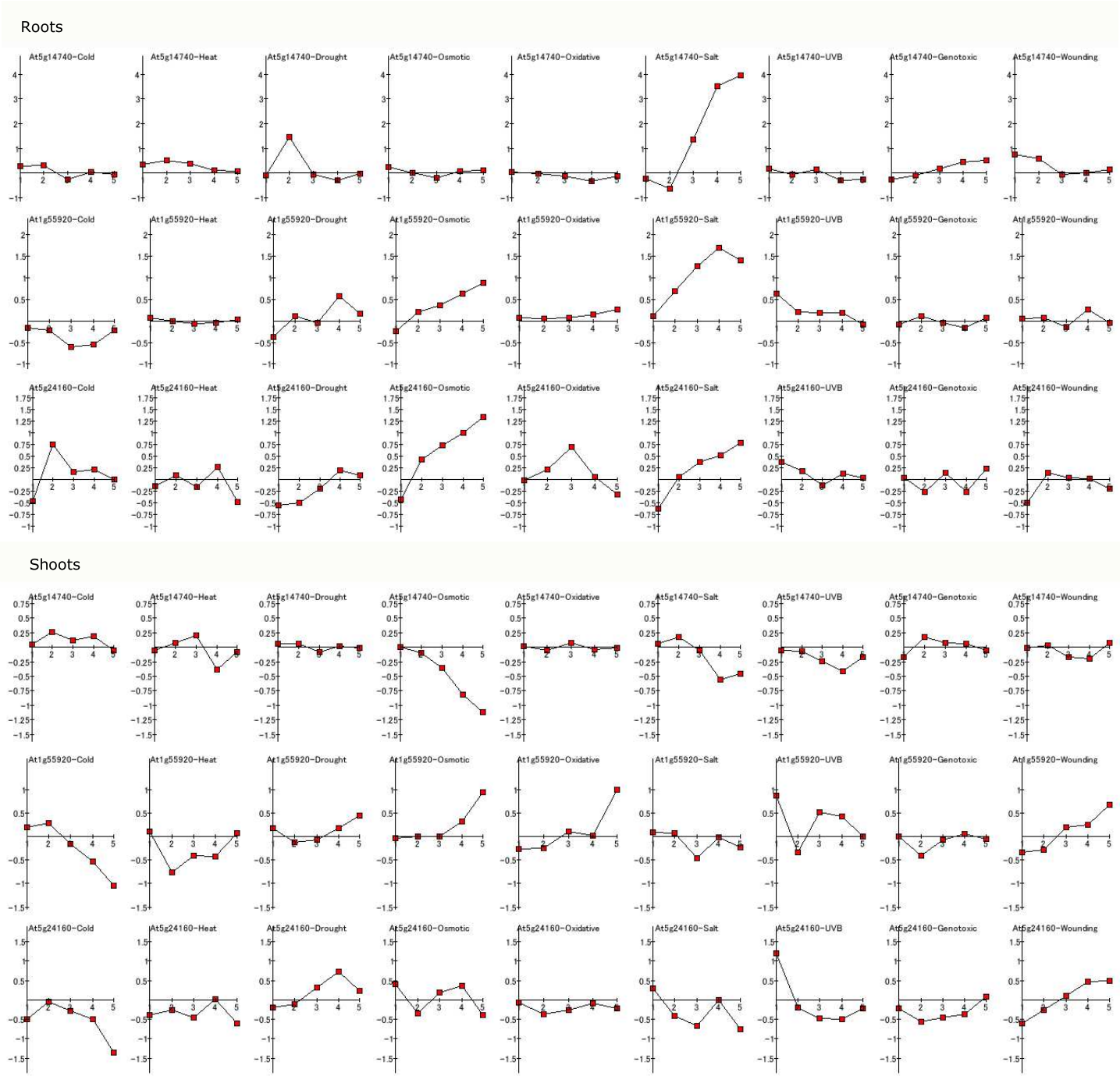


Figure 8.11: Expression vectors for genes in Itemset 12

Table 8.10: Selected Significant motif combinations from senescence genes in shoots

	At1g09500	At1g73160	At4g37990	At4g39090	At5g59310
ARFAT					
DPBFCOREDCDC3					
GT1CONSENSUS					
MYCCONSENSUSAT					
RAV1AAT					
ZAT12-down					
IBOX					
Fed-AtMyb4					
GAREAT					
MRE					
MYB1LEPR					
MYB2CONSENSUSAT					
WBOXATNPR1					
Key	 :Itemset 24, $p = 1.495e-05$		:Itemset 32, $p = 4.746e-05$		

Chapter 9

Conclusions and Future Directions

Using both itemset significance data and gene expression correlation data for genes ensures that the biological context is represented in the final p-value calculated. XcisClique identifies new motif and gene combinations that might indicate the involvement of that set of genes in biological functions and processes they are not annotated with. Given a set of genes known to be tightly correlated in the context of a biological function, correlating this set to the rest of the genes of the genome would result in an enriched set whose analysis would result in more itemsets of such a type. Also, conserved arrangements of motifs were observed in significant itemsets. The current XcisClique system does not address ordered arrangements of *cis*-elements. However, itemsets are logical candidates to scan for conserved motif order and separation, since identifying the most significant itemsets greatly reduces the number of combinations to be scanned for motif order in a given input set of genes. In yeast, it has been observed that the arrangement of *cis*-elements is biologically important for transcription factor activity in some genes and not in others [Terai and Takagi, 2004]. If this concept is extended to *Arabidopsis thaliana*, each itemset can be further analyzed to get a measure of how conserved the arrangement of a set of motifs is in a set of genes. This can be done by modeling the arrangement of motifs either as directed graphs or suffix trees. A formalization of the process to identify conserved arrangements is one of the future directions. Conservation of order of smaller modules of *cis*-elements in the promoter of a gene, also makes more biological sense than conservation of order of all the *cis*-elements present in a gene promoter.

In every significant itemset, separations between pairs of *cis*-elements can be analyzed to detect

conserved distances. The most conserved pairs can then serve as modules, distances between which can be analyzed. This process can be recursively repeated until a threshold is reached.

One of the drawbacks of XcisClique is that patterns are located by exact pattern matching. Using Perl regular expressions makes the matching process flexible. However, the matching process could be augmented with a scoring system that penalizes mutations on a position-sensitive basis. Also, unlike in yeast, not all *Arabidopsis thaliana* transcription factor binding sites are known. The present collection of *Arabidopsis thaliana cis*-elements in XcisClique has been curated from PLACE and the literature. But many more regulatory elements remain to be discovered. Existing motif discovery methods such as those described in Section 3.2 can serve as methods of discovering putative TFBSs, which can then be used as XcisClique inputs.

A number of challenging computational problems exist in the field of modeling and analysis of eukaryotic promoters. Modeling of *cis*-elements is an interesting problem in the absence of adequate data to construct position weight matrices. While regular expressions can represent a set of strings efficiently, they cannot model the probabilities of occurrence of nucleotides at each position of a *cis*-element. An ideal method of modeling a *cis*-element could be regular expressions with PWM probabilities built into them.

Bibliography

Frank Andel, Andreas G. Ladurner, Carla Inouye, Robert Tjian, and Eva Nogales. Three-Dimensional Structure of the Human TFIID-IIA-IIB Complex. *Science*, 286:2153–2156, 1999.

Vladimir B. Bajic, Seng Hong Seah, Allen Chong, Guanglan Zhang, Judice L. Y. Koh, and Vladimir Brusiv. Dragon Promoter Finder: recognition of vertebrate RNA polymerase II promoters. *Bioinformatics*, 18:198–199, 2002.

SS Baker, KS Wilhelm, and MF Thomashaw. The 5'-region of *Arabidopsis thaliana* cor15a has *cis*-acting elements that confer cold-, drought- and ABA-regulated gene expression. *Plant Mol. Biol.*, 24:701–13, Mar 1994.

MD Barros, E Czarnecka, and WB Gurley. Mutational analysis of a plant heat shock element. *Plant Molecular Biology*, 19:665–75, Jul 1992.

Jeremy M. Berg, John L. Tymoczko, and Lubert Stryer. *Biochemistry.* W.H. Freeman and Company, NCBI, 5th edition, 2002.

M Bienz and HR Pelham. Mechanisms of heat-shock gene activation in higher eukaryotes. *Adv Genet*, 24:31–72, 1987.

Robert J. Brooker. *Genetics: Analysis and Principles.* Addison Wesley Longman, Inc, 1999.

Sylvain La Camera, Guillaume Gouzerh, Sandrine Dhondt, Laurent Hoffmann, Bernard Fritig, Michel Legrand, and Thierry Heitz. Metabolic reprogramming in plant innate immunity: the contributions of phenylpropanoid and oxylipin pathways. *Immunological Reviews*, 198:267–284, 2004.

D Cao, JE Froehlich, H Zhang, and CL Cheng. The chlorate-resistant and photomorphogenesis-defective mutant cr88 encodes a chloroplast-targeted HSP90. *Plant J*, 33:107–118, 2003.

Derek Y Chiang, Alan B Moses, Manolis Kellis, Eric S Lander, and Michael B Eisen. Phylogenetically and spatially conserved word pairs associated with gene-expression changes in yeast. *Genome Biology*, 4(R43), June 2003.

Chan CS, Guo L, and Shih MC. Promoter analysis of the nuclear gene encoding the chloroplast glyceraldehyde-3-phosphate dehydrogenase B subunit of *Arabidopsis thaliana*. *Plant Mol Biol*, 46: 131–141, 2001.

E Czarnecka-Verner, S Pan, T Salem, and WB Gurley. Plant class B HSFs inhibit transcription and exhibit affinity for TFIIB and TBP. *Plant Mol Biol.*, 56:57–75, 2004.

James W. Fickett and Artemis G. Hatzigeorgiou. Eukaryotic Promoter Recognition. *Genome Research*, 7:861–878, 1997.

Sarah Fowler and MF Thomashow. *Arabidopsis* transcriptome profiling indicates that multiple regulatory pathways are activated during cold acclimation in addition to the CBF cold response pathway. *Plant Cell*, 14:1675–1690, 2002.

Sarah G. Fowler, Daniel Cook, and Michael F. Thomashow. Low temperature induction of *Arabidopsis* CBF1, 2, and 3 is gated by the circadian clock. *Plant Physiol.*, 137(3):961–968, 2005. URL `http://www.plantphysiol.org/cgi/content/abstract/137/3/961`.

Shimon Gepstein, Gazalah Sabehi, Marie-Jeanna Carp, Taleb Hajouj, Mizied Falah Orna Nesher, Inbal Yariv, Chen Dor, and Michal Bassani. Large-scale identification of leaf senescence-associated genes. *The Plant Journal*, 36:629–642, 2003.

Anil Grover, Anvish Kapoor, O. Satya Lakshmi, Sangeeta Agarwal, Chandan Sahi, Surekha Katiyar-Agarwal, Manu Agarwal, and Himanshu Dubey. Understanding molecular alphabets of the plant abiotic stress responses. *Current Science*, 80(2), January 2001.

Mayetri Gupta and Jun S. Liu. *De novo cis*-regulatory module elicitation for eukaryotic genomes. *PNAS*, 102(20), May 2005.

Kosmas Haralampidis, Dimitri Milioni, Stamatis Rigas, and Polydefkis Hatzopoulos. Combinatorial interaction of *cis*-elements specifies the expression of the *Arabidopsis* AtHsp90-1 Gene. *Plant Physiology*, 129:1138–1149, July 2002.

SL Harmer, JB Hogenesch, M Straume, HS Chang, B Han, T Zhu, X Wang, JA Kreps, and SA Kay. Orchestrated transcription of key pathways in *Arabidopsis* by the circadian clock. *Science*, 290, 2000.

Daniel L. Hartl and Elizabeth W. Jones. *Essential Genetics: A Genomics Perspective*. Jones and Bartlett, 3rd edition, 2002.

Z He, L Li, and S Luan. Immunophilins and parvulins. Superfamily of peptidyl prolyl isomerases in *Arabidopsis*. *Plant Physiol*, 134:1248–1267, 2004.

K Higo, Y Ugawa, M Iwamoto, and T Korenaga. Plant *cis*-acting regulatory DNA elements(PLACE) database. *Nucleic Acids Research*, 27(1):297–300, 1999.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley Longman, 2nd edition, 2001.

DA Hubert, P Tornero, Y Belkhadir, P Krishna, A Takahashi, K Shirasu, and JL Dangl. Cytosolic HSP90 associates with and modulates the *Arabidopsis* RPM1 disease resistance protein. *EMBO J*, 22, 2003.

Matthew E. Hudson and Peter H. Quail. Identification of promoter motifs involved in the network of phytochrome A-regulated gene expression by combined analysis of genomic sequence and microarray data. *Plant Physiology*, 133:1605–1616, December 2003.

GB Hutchinson. The prediction of vertebrate promoter regions using different hexamer frequency analysis. *Comput Appl Biosci*, 1996.

S Ishiguro, Y Watanabe, N Ito, H Nonaka, N Takeda, T Sakai, H Kanaya, and K Okada. SHEPHERD is the *Arabidopsis* GRP94 responsible for the formation of functional CLAVATA proteins. *Embo J*, 21:898–908, 2002.

Lars Juhl Jensen and Steen Knudsen. Automatic discovery of regulatory patterns in promoter regions based on whole cell expression data and functional annotation. *Bioinformatics*, 16(4):326–333, April 2000.

Alan F. Karr. *Probability*. Springer-Verlag, 1993.

Mamoru Kato, Naoya Hata, Nilanjana Banerjee, Bruce Futcher, and Michael Q Zhang. Identifying combinatorial regulation of transcription factors and binding motifs. *Genome Biology*, 5(R56), July 2004.

A.E. Kel, E. Gobling, I. Reuter, E. Cheremushkin, O.V. Kel-Margoulis, and E. Wingender. MATCH: a tool for searching transcription factor binding sites in DNA sequences. *Nucleic Acids Research*, 31(13):3576–3579, 2003.

T Kiyosue, JK Beetham, F Pinot, BD Hammock, K Yamaguchi-Shinozaki, and K Shinozaki. Characterization of an *Arabidopsis* cDNA for a soluble epoxyde hydrolase gene that is inducible by auxin and water stress. *Plant J*, 6, 1994.

P Krishna, M Sacco, JF Cherutti, and S Hill. Cold-induced accumulation of HSP90 transcripts in *Brassica napus*. *Plant Physiol*, 107:915–923, 1995.

Priti Krishna and Greg Gloor. The HSP90 family of proteins in *Arabidopsis thaliana*. *Cell Stress & Chaperones*, 6(3):238–246, 2001.

C Laloi, K Apel, and A Danon. Reactive oxygen signalling: the latest news. *Curr Opin Plant Biol.*, 7:323–8, 2004.

Charles E. Lawrence, Stephen F. AltSchul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, October 1993.

J. M. Lelievre, L. O. Oliveira, and N. C. Nielsen. 5'-CATGCAT-3' elements modulate the expression of glycinin genes. *Plant Physiology*, 98:387–391, 1992.

Harry R. Lewis and Christos H. Papadimitriou. *Elements of The Theory of Computation*. Pearson Education Pte. Ltd., 2002.

Bai-Ling Lin, Jang-Shiun Wang, Hung-Chi Liu, Rung-Wu Chen, Yves Meyer, Abdellalli Barakat, and Michel Delseny. Genomic analysis of the HSP70 superfamily in *Arabidopsis thaliana*. *Cell Stress & Chaperones*, 3:201–208, 2001.

R Mahalingam, A Gomez-Buitrago, N Eckardt, N Shah, A Guevara-Garcia, P Day, R Raina, and NV Fedoroff. Characterizing the stress/defense transcriptome of *Arabidopsis*. *Genome Biology*, 4, 2003.

IM Martinez and MJ Chrispeels. Genomic analysis of the unfolded protein response in *Arabidopsis* shows its connection to important cellular processes. *Plant Cell*, 15:561–576, 2003.

Jan A Miernyk. The J-domain proteins of *Arabidopsis thaliana*: an unexpectedly large and diverse family of chaperones. *Cell Stress and Chaperones*, 6:209–218, 2001.

D Milioni and P Hatzopoulos. Genomic organization of HSP90 gene family in *Arabidopsis*. *Plant Mol Biol*, 35:955–961, 1997.

Julie Nardone, Dong U Lee, K Mark Ansel, and Anjana Rao. Bioinformatics for the 'bench biologist': how to find regulatory regions in genomic DNA. *Nature Immunology*, 5, August 2004.

NCBI Expression Resources. http://www.ncbi.nlm.nih.gov/Class/NAWBIS/Modules/Expression/exp14.html.

Lutz Nover, Kapil Bharti, Shravan Kumar Mishra, Arnab Ganguli, and Klaus-Dieter Scharf. *Arabidopsis* and the heat stress transcription factor world: how many heat stress transcription factors do we need? *Cell Stress & Chaperones*, 6(3):177–189, 2001.

Lutz Nover, D Newmann, U zur Nieden, R Manteuffel, G walter, and K D Scharf. Intracellular localization of heat shock protein in tomato cell cultures. *Eur Journal of Cell Biology*, 43:71–81, 1987.

M Ohgishi, A Oka, G Morelli, I Ruberti, and T Aoyama. Negative autoregulation of the *Arabidopsis* homeobox gene ATHB-2. *Plant J*, 25:389–398, 2001.

Uwe Ohler. *Computational Promoter Recognition in Eukaryotic Genomic DNA*. PhD thesis, Universitat Erlangen-Nurnberg, 2001.

S Persson, M Rosenquist, K Svensson, R Galvao, WF Boss, and M Sommarin. Phylogenetic analyses and expression studies reveal two distinct groups of calreticulin isoforms in higher plants. *Plant Physiol*, 133:1385–1396, 2003.

Yitzhak Pilpel, Priya Sudarsanam, and George M. Church. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nature Genetics*, September 2001.

C Prasinos, K Krampis, D Samakovli, and P Hatzopoulos. Tight regulation of expression of two *Arabidopsis* cytosolic HSP90 genes during embryo development. *J Expt Biol*, 56:633–644, 2005.

WB Pratt, P Krishna, and LJ Olsen. HSP90-binding immunophilins in plants: the protein movers. *Trends Plant Sci*, 6:54–58, 2001.

Viviane Praz, Rouaida Perier, Claude Bonnard, and Philipp Bucher. The Eukaryotic Promoter Database, EPD: new entry types and links to gene expression data. *Nucleic Acids Research*, 30 (1):322–324, 2002.

Dan S. Prestridge. Predicting Pol II promoter sequences using transcription factor binding sites. *Journal of Molecular Biology*, 249:923–932, 1995.

Kerstin Quandt, Kornelie Frech, Holger Karas, Edgar Wingender, and Thomas Werner. MatInd and MatInspector: new fast and versatile tools for detection of consensus matches in nucleotide sequence data. *Nucleic Acids Research*, 23(23):4878–4884, 1995.

M. G. Reese and F. H. Eeckman. Novel neural network algorithms for improved eukaryotic promoter site recognition, 1995.

Jose Luis Riechmann. *Transcriptional Regulation: a Genomic Overview. The Arabidopsis Book*. American Society of Plant Biologists, 2002.

Stephane Rombauts, Kobe Florquin, Magali Lescot, Kathleen Marchal, Pierre Rouze, and Yves Van de Peer. Computational approaches to identify promoters and *cis*-regulatory elements in plant genomes. *Plant Physiology*, 132:1162–1176, 2003.

Frederick P. Roth, Jason D. Hughes, Preston W. Estep, and George M. Church. Finding DNA regulatory motifs within unaligned non-coding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology*, 16, October 1998.

Y Sakuma, Q Liu, JG Dubouzet, H Abe, K Shinozaki, and K Yamaguchi-Shinozaki. DNA-binding specificity of the ERF/AP2 domain of *Arabidopsis* DREBs, transcription factors involved in dehydration- and cold-inducible gene expression. *Biochemical and Biophysical Research Communications*, 290:998–1009, 2002.

Albin Sandelin, Pär Engström Wynand Alkema, Wyeth Wasserman, and Boris Lenhard. JASPAR: an open access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Research*, 32, 2004.

Nicholas Santoro, Nina Johansson, and Dennis J. Thiele. Heat shock element architecture is an important determinant in the temperature and transactivation domain requirements for heat shock transcription factor. *Molecular and Cellular Biology*, 18(11):6340–6352, November 1998.

Eric Sayers and David Wheeler. *Building Customized Data Pipelines Using the Entrez Programming Utilities (eUtils)*. NLM, 2003.

Klaus-Dieter Scharf, Masood Siddique, and Elixabeth Vierling. The expanding family of *Arabidopsis thaliana* small heat stress proteins and a new family of proteins containing α-crystallin domains (Acd proteins). *Cell Stress & Chaperones*, 6(3):225–237, 2001.

Fritz Schoffl, Ralf Prandl, and Andreas Reindl. Regulation of the heat shock response. *Plant Physiology*, 117:1135–1141, 1998.

R M Schrode and RJ Kaufman. ER stress and the unfolded protein response. *Mutat Res*, 569:29–63, 2005.

Jonathan Schug and G. Christian Overton. TESS: Transcription Element Search System on the WWW. Technical Report CBIL-TR-1997-1001-v0.0, University of Pennsylvania, 1998.

I. A. Shahmuradov, V. V. Solovyev, and A. J. Gammerman. Plant promoter prediction with confidence estimation. *Nucleic Acids Research*, 33:1069–1076, 2005.

Qingxi Shen and Tuan-Hua David Ho. Functional dissection of an abscissic acid (ABA)-inducible gene reveals two independent ABA-responsive complexes each containing a G-box and a novel *cis*-acting element. *The Plant Cell*, 7:295–307, March 1995.

David J. Sheskin. *Parametric and Non-parametric Statistical Procedures*. Chapman and Hall/CRC, 2nd edition, 2000.

Saurabh Sinha and Martin Tompa. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 30(24):5549–5560, September 2002.

John D. Storey and Robert Tibshirani. Statistical significance for genomewide studies. *PNAS*, 100 (16):9440–9445, 2003.

T Taji, C Ohsumi, S Iuchi, M Seki, M Kasuga, M Kobayashi, K Yamaguchi-Shinozaki, and K Shinozaki. Important roles of drought- and cold-inducible genes for galactinol synthase in stress tolerance in *Arabidopsis thaliana*. *Plant J.*, 29:417–26, 2002.

A Takahashi, C Casais, K Ichimura, and K Shirasu. HSP90 interacts with RAR1 and SGT1 and is essential for RPS2-mediated disease resistance in *Arabidopsis*. *Proc Natl Acad Sci*, 100:11777–11782, 2003.

W Tang and SE Perry. Binding site selection for the plant MADS domain protein AGL15: an *in vitro* and *in vivo* study. *J. Biol Chem*, 278:28154–28159, 2003.

Goro Terai and Toshihisa Takagi. Predicting rules on organization of *cis*-regulatory elements, taking the order of elements into account. *Bioinformatics*, 20(7):1119–1128, February 2004.

Gert Thijs, Kathleen Marchal, Magali Lescot, Stephane Rombauts, Bart de Moor, Pierre Rouze, and Yves Moreau. A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *Journal of Computational Biology*, 9(2):447–464, 2002.

Robert Tjian. Molecular machines that control genes. *Scientific American*, 272(2):54–61, February 1995.

J. van Helden, B. Andre, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281:827–842, 1998.

JT Vogel, DG Zarka, HA van Buskirk, SG Fowler, and MF Thomashow. Roles of the CBF2 and ZAT12 transcription factors in configuring the low temperature transcriptome of *Arabidopsis*. *Plant Journal*, 41:105–211, 2004.

ZY Wang, D Kenigsbuch, L Sun, E Harel, MS Ong, and EM Tobin. A MYB-related transcription factor is involved in the phytochrome regulation of an *Arabidopsis* Lhcb gene. *Plant Cell*, 9: 491–50, 1997.

Wyeth W. Wasserman and Albin Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nature Genetics*, 5, April 2004.

Robert F. Weaver and Philip W. Hedrick. *Genetics*. Wm. C. Brown Publishers, 1997.

Thomas Werner. Models for prediction and recognition of eukaryotic promoters. *Mammalian Genome, Incorporating Mouse Genome*, 10:168–175, 1999.

A. Wingender, E.and Kel, H. Kel, O.and Karas, T. Heinemeyer, P. Dietze, R. Knuppel, A. Romaschenko, and N. Kolchanov. TRANSFAC, TRRD and COMPEL: towards a federated database system on transcriptional regulation. *Nucleic Acids Research*, 25:265–268, 1997.

Eric P. Xing, Wei Wu, Michael I. Jordan, and Richard M. Karp. LOGOS: a modular Bayesian model for *de novo* motif detection. *Journal of Bioinformatics and Computational Biology*, 2(1):127–154, 2004.

N Yabe, T Takahashi, and Y Komeda. Analysis of tissue-specific expression of *Arabidopsis thaliana* HSP90-family gene HSP81. *Plant Cell Physiol*, 35:1207–1219, 1994.

D Yu, C Chen, and Z Chen. Evidence for an important role of WRKY DNA binding proteins in the regulation of NPR1 gene expression. *Plant Cell*, 13, 2001.

Z Zhang, MK Quick, KC Kanelakis, M Gijzen, and P Krishna. Characterization of a plant homolog of HOP, a cochaperone of HSP90. *Plant Physiol.*, 131:525–535, 2003.

JK Zhu. Salt and drought stress signal transduction in plants. *Annu Rev Plant Biol.*, 53:247–73, 2002.

Appendix A

cis-Elements used in XcisClique

Table A.1: List of Regulatory Elements used in XcisClique

***Cis*-Element**	**Pattern**
HSE_Perfect	(GAA[ACTG]{2}TTC[ACTG]{2}GAA[ACTG]{2})|
	(TTC[ACTG]{2}GAA[ACTG]{2}TTC[ACTG]{2})|
	(GAA[ACTG]{2}GAA[ACTG]{2}GAA[ACTG]{2})|
	(TTC[ACTG]{2}TTC[ACTG]{2}TTC[ACTG]{2})
HSE_1-mut	(GA[CGT][ACGT]{2}TTC[ACTG]{2}GAA)|
	(G[CGT]A[ACGT]{2}TTC[ACTG]{2}GAA)|
	(GAA[ACGT]{2}[ACG]TC[ACTG]{2}GAA)|
	(GAA[ACGT]{2}T[ACG]C[ACTG]{2}GAA)|
	(GAA[ACGT]{2}TTC[ACTG]{2}G[CGT]A)|
	(GAA[ACGT]{2}TTC[ACTG]{2}GA[CGT])|
	([ACG]TC[ACGT]{2}GAA[ACGT]{2}TTC)|
	(T[ACG]C[ACGT]{2}GAA[ACGT]{2}TTC)|
	(TTC[ACGT]{2}G[CGT]A[ACGT]{2}TTC)|
	(TTC[ACGT]{2}GA[CGT][ACGT]{2}TTC)|
	(TTC[ACGT]{2}GAA[ACGT]{2}[ACG]TC)|
	(TTC[ACGT]{2}GAA[ACGT]{2}T[ACG]C)|
	(CTNGAANNTTCNA)
HSE_2-mut	(GAA[ACGT]{2}[ACG]TC[ACGT]{2}GA[CGT])|
	(GAA[ACGT]{2}T[ACG]C[ACGT]{2}G[CGT]A)|
	(GAA[ACGT]{2}T[ACG]C[ACGT]{2}GA[CGT])|
	(G[CGT]A[ACGT]{2}[ACG]TC[ACTG]{2}GAA)|

	(G[CGT]A[ACGT]{2}T[ACG]C[ACTG]{2}GAA)|
	(G[CGT]A[ACGT]{2}TTC[ACGT]{2}G[CGT]A)|
	(G[CGT]A[ACGT]{2}TTC[ACGT]{2}GA[CGT])|
	(GA[CGT][ACGT]{2}[ACG]TC[ACTG]{2}GAA)|
	(GA[CGT][ACGT]{2}T[ACG]C[ACTG]{2}GAA)|
	(GA[CGT][ACGT]{2}TTC[ACGT]{2}G[CGT]A)|
	(GA[CGT][ACGT]{2}TTC[ACGT]{2}GA[CGT])|
	(GAA[ACGT]{2}[ACG]TC[ACGT]{2}G[CGT]A)|
	(TTC[ACGT]{2}G[CGT]A[ACGT]{2}T[ACG]C)|
	(TTC[ACGT]{2}GA[CGT][ACGT]{2}[ACG]TC)|
	(TTC[ACGT]{2}GA[CGT][ACGT]{2}T[ACG]C)|
	([ACG]TC[ACGT]{2}G[CGT]A[ACGT]{2}TTC)|
	([ACG]TC[ACGT]{2}GA[CGT][ACGT]{2}TTC)|
	([ACG]TC[ACGT]{2}GAA[ACGT]{2}[ACG]TC)|
	([ACG]TC[ACGT]{2}GAA[ACGT]{2}T[ACG]C)|
	(T[ACG]C[ACGT]{2}G[CGT]A[ACGT]{2}TTC)|
	(T[ACG]C[ACGT]{2}GA[CGT][ACGT]{2}TTC)|
	(T[ACG]C[ACGT]{2}GAA[ACGT]{2}[ACG]TC)|
	(T[ACG]C[ACGT]{2}GAA[ACGT]{2}T[ACG]C)|
	(TTC[ACGT]{2}G[CGT]A[ACGT]{2}[ACG]TC)
14BPATERD1	CACTAAATTGTCAC
AAGACGTAGATACL12	AAGACGTAG
ABREATCONSENSUS	YACGTGGC
ABREATRD22	RYACGTGGYR
ABRELATERD1	ACGTG
ABREZMRAB28	CCACGTGG
ACEATCHS	GACACGTAGA
ACGTABREMOTIFA2OSEM	ACGTGKC
AGAMOUSATCONSENSUS	TTDCCWWWWWWGGHAA
AGATCONSENSUS	TTWCCWWWWNNGGWW
AGCBOXNPGLB	AGCCGCC
AGL1ATCONSENSUS	NTTDCCWWWWNNGGWAAN
AGL2ATCONSENSUS	NNWNCCAWWWWTRGWWAN
AGL3ATCONSENSUS	TTWCYAWWWWTRGWAA
AP1	ATGACTCA
ARFAT	TGTCTC
ASF1MOTIFCAMV	TGACG
ATHB1ATCONSENSUS	CAATWATTG

ATHB2ATCONSENSUS	CAATSATTG
ATHB5ATCORE	CAATNATTG
ATHB6COREAT	CAATTATTA
C/EBP	CCAAT
CACGTGMOTIF	CACGTG
CARG1ATAP3	GTTTACATAAATGGAAAA
CARG2ATAP3	CTTACCTTTCATGGATTA
CARG3ATAP3	CTTTCCATTTTTAGTAAC
CARGATCONSENSUS	CCWWWWWWGG
CARGNCAT	CCWWWWWWWWGG
CBF2-down	(AGCNCGNCT)|(CACMACAC)|(CAAGTTGR)| (CGAWCYAG)|(CTTTGCCT)|(TTCNGAGT)
CCA1ATLHCB1	AAMAATCT
CDA1ATCAB2	CAAAACGC
CGF1ATCAB2	GATAAAGATTACTTCAGATATAACAAACGTTAC
DPBFCOREDCDC3	(ACACCCG)| (CCACTTG)| (ACACGCG)| (ACACGTG)
DRECRTCOREAT	RCCGAC
DREDR1ATRD29AB	TACCGACAT
E2FANTRNR	TTTCCCGC
E2FAT	TYTCCCGCC
E2FATLIKE	(ATTCCCGCC)|(GTTCGCGCC)|(GTTGGCGCC)| (GTTGGCGGG)|(TCTCGCGCC)| (TCTCGCGGCCC)| (TCTGCCGC)|(TCTGGCGC)|(TCTGGCGGC)| (TTCTGGCGC)|(TTTCCCGC)|(ATTCGCGC)| (TTTCCCGCG)|(TTTCCGGCG)| (TTTCCGGCGG)| (TTTCGCC)|(TTTCGCGCG)|(TTTCGCGGC)| (TTTCGGCC)|(TTTCGGCCC)|(TTTCGGCGG)| (TTTCGGGCC)|(ATTCGCGCG)|(TTTGCCGCC)| (TTTGCCGCG)|(TTTGCCGG)|(TTTGCCGGC)| (TTTGGCCG)|(TTTGGCCGG)|(TTTGGCGC)| (TTTGGCGCC)|(TTTGGCGG)|(TTTGGCGGC)| (ATTGCGCC)|(TTTGGCGGG)|(TTTGGGCCG)| (TTTCCCGGC)|(TTTCCCGGG)|(TTTCGCGCC)| (TTTCGCGGG)|(TTTGCCGGG)|(TTTGGCGCG)| (ATTGCGCGG)|(ATTGGCGGG)|(CTTCCCGCC)|

	(CTTCGCGCC)| (GTTCCCGCC)
EIN3ATERF1	GGATTCAAGGGGCATGTATCTTGAATCC
ELRECOREPCRP1	TTGACC
EMBP1TAEM	CACGTGGC
EREGCC	TAAGAGCCGCC
EVENINGAT	AAAATATCT
Fed-ABRE-like	BACGTGKM
Fed-AtMyb4	AMCWAMC
Fed-GBF	CACGTG
Fed-HBF	CCTACC
Fed-SA-induced	ACGTCA
GADOWNAT	ACGTGTC
GAREAT	TAACAAR
GBOXLERBCS	MCACGTGGC
GCCCORE	GCCGCC
GT1CONSENSUS	GRWAAW
HDZIP2ATATHB2	TAATMATTA
HEXAMERATH4	CCGTCG
HEXAT	TGACGTGG
HY5AT	TGACACGTGGCA
IBOX	GATAAG
IDRSZMFER1	CACGAGSCCKCCAC
JASE1ATOPR1	CGTCAATGAA
JASE2ATOPR1	CATACGTCGTCAA
L1BOXATPDF1	TAAATGYA
LEAFYATAG	CCAATGT
LREBOX2PSRBCS3	TGTGTGGTTAATATG
LS5ATPR1	ACGTCATAGA
LS7ATPR1	TCTACGTCAC
LTREATLTI78	ACCGACA
LTRECOREATCOR15	CCGAC
MRE	(TGCRCNC)| (TGCGCAAC)| (TGCAGAC)
MREATCHS	TCTAACCTACCA
MYB1AT	WAACCA
MYB1LEPR	GTTAGTT
MYB2AT	TAACTG
MYB2CONSENSUSAT	YAACKG
MYBATRD22	CTAACCA

MYBCORE	CNGTTR
MYBPLANT	MACCWAMC
MYCATERD1	CATGTG
MYCATRD22	CACATG
MYCCONSENSUSAT	CANNTG
NONAMERATH4	AGATCGACG
OBF5ATGST6	ATCTTATGTCATTGATGACGACCTCC
OBP1ATGST6	TACACTTTTGG
OCSELEMENTAT	TGACGYAAGSRMTKACGYMM
OCTAMERMOTIFTAH3H4	CGCGGATC
PIATGAPB	GTGATCAC
PIIATGAPB	TTGGTTTTGATCAAAACCAA
RAV1AAT	CAACA
RAV1BAT	CACCTG
RYREPEATVFLEB4	CATGCATG
STRE	CCCCT
SV40COREENHAN	GTGGWWHG
TBOXATGAPB	ACTTTG
TEF1BOXATA1	ACAGGGGCATAATGGTAATTTAAA
TEFBOXATEEF1AA1	AGGGGCATAATGGTAA
TELOBOXATEEF1AA1	AAACCCTAA
UPRE1AT	ATTGGTCCACG
UPRE2AT	CCACGTCATC
UPRMOTIFIAT	CCACGTCA
UPRMOTIFIIAT	CCNNNNNNNNNNNCCACG
WBOXATNPR1	TTGAC
WUSATAg	TTAATGG
ZDNAFORMINGATCAB1	ATACGTGT
ZAT12-up	(TCSNCTCS)|(GCATTGAC)|(YCTCTTCA)| (CAATGMKG)|(TGAGGTCA)|(RSAATGAG)
ZAT12-down	(MCAACTTS)|(AWGAKGWC)|(ACAWMTTC)| (TATCCAAA)|(GAMTAAGA)|(TAYGCCAG)

Appendix B

Slides used for retrieving Expression Data

The slides used in the following postgreSQL query have been used to retrieve gene expression data for XcisClique.

```
\f '\t'
\a
\t
select A.at_id,D.probe_id,D.slide_id,D.logi from at_to_affy A,arraydata D
where (A.affy_id = D.probe_id) and
(D.slide_id like '%AtGen_6-0111_Control-Shoots-0.5h_Rep1%'
 or D.slide_id like '%AtGen_6-0112_Control-Shoots-0.5h_Rep2%'
 or D.slide_id like '%AtGen_6-0121_Control-Roots-0.5h_Rep1%'
 or D.slide_id like '%AtGen_6-0122_Control-Roots-0.5h_Rep2%'
 or D.slide_id like '%AtGen_6-0211_Control-Shoots-1.0h_Rep1%'
 or D.slide_id like '%AtGen_6-0212_Control-Shoots-1.0h_Rep2%'
 or D.slide_id like '%AtGen_6-0221_Control-Roots-1.0h_Rep1%'
 or D.slide_id like '%AtGen_6-0222_Control-Roots-1.0h_Rep2%'
 or D.slide_id like '%AtGen_6-0311_Control-Shoots-3.0h_Rep1%'
 or D.slide_id like '%AtGen_6-0312_Control-Shoots-3.0h_Rep2%'
```

```
or D.slide_id like '%AtGen_6-0321_Control-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-0322_Control-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-0411_Control-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-0412_Control-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-0421_Control-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-0422_Control-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-0511_Control-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-0512_Control-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-0521_Control-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-0522_Control-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-3111_Saltstress-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-3112_Saltstress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-3121_Saltstress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-3122_Saltstress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-3211_Saltstress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-3212_Saltstress-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-3221_Saltstress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-3222_Saltstress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-3311_Saltstress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-3312_Saltstress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-3321_Saltstress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-3322_Saltstress-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-3411_Saltstress-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-3412_Saltstress-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-3421_Saltstress-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-3422_Saltstress-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-3511_Saltstress-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-3512_Saltstress-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-3521_Saltstress-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-3522_Saltstress-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-4111_Droughtstress-Shoots-0.5h_Rep1%'
```

```
or D.slide_id like '%AtGen_6-4112_Droughtstress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-4121_Droughtstress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-4122_Droughtstress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-4211_Droughtstress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-4212_Droughtstress-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-4221_Droughtstress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-4222_Droughtstress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-4311_Droughtstress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-4312_Droughtstress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-4321_Droughtstress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-4322_Droughtstress-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-4411_Droughtstress-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-4412_Droughtstress-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-4421_Droughtstress-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-4422_Droughtstress-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-4511_Droughtstress-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-4512_Droughtstress-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-4521_Droughtstress-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-4522_Droughtstress-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-5111_Genotoxicstress-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-5112_Genotoxicstress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-5121_Genotoxicstress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-5122_Genotoxicstress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-5211_Genotoxicstress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-5212_Genotoxicstress-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-5221_Genotoxicstress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-5222_Genotoxicstress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-5311_Genotoxicstress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-5312_Genotoxicstress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-5321_Genotoxicstress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-5322_Genotoxicstress-Roots-3.0h_Rep2%'
```

```
or D.slide_id like '%AtGen_6-5411_Genotoxicstress-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-5412_Genotoxicstress-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-5421_Genotoxicstress-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-5422_Genotoxicstress-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-5511_Genotoxicstress-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-5512_Genotoxicstress-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-5521_Genotoxicstress-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-5522_Genotoxicstress-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-6111_Oxidativestress-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-6112_Oxidativestress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-6124_Oxidativestress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-6122_Oxidativestress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-6211_Oxidativestress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-6212_Oxidativestress-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-6223_Oxidativestress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-6224_Oxidativestress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-6311_Oxidativestress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-6312_Oxidativestress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-6323_Oxidativestress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-6322_Oxidativestress-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-6411_Oxidativestress-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-6412_Oxidativestress-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-6421_Oxidativestress-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-6422_Oxidativestress-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-6511_Oxidativestress-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-6512_Oxidativestress-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-6523_Oxidativestress-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-6524_Oxidativestress-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-7711_UV-Bstress-Shoots-0.25h_Rep1%'
or D.slide_id like '%AtGen_6-7712_UV-Bstress-Shoots-0.25h_Rep2%'
or D.slide_id like '%AtGen_6-7721_UV-Bstress-Roots-0.25h_Rep1%'
```

```
or D.slide_id like '%AtGen_6-7722_UV-Bstress-Roots-0.25h_Rep2%'
or D.slide_id like '%AtGen_6-7111_UV-Bstress-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-7112_UV-Bstress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-7121_UV-Bstress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-7122_UV-Bstress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-7211_UV-Bstress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-7212_UV-Bstress-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-7221_UV-Bstress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-7222_UV-Bstress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-7311_UV-Bstress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-7312_UV-Bstress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-7321_UV-Bstress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-7322_UV-Bstress-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-7411_UV-Bstress-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-7412_UV-Bstress-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-7421_UV-Bstress-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-7422_UV-Bstress-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-7511_UV-Bstress-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-7512_UV-Bstress-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-7521_UV-Bstress-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-7522_UV-Bstress-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-7611_UV-Bstress-Shoots-24.0h_Rep1%'
or D.slide_id like '%AtGen_6-7612_UV-Bstress-Shoots-24.0h_Rep2%'
or D.slide_id like '%AtGen_6-7621_UV-Bstress-Roots-24.0h_Rep1%'
or D.slide_id like '%AtGen_6-7622_UV-Bstress-Roots-24.0h_Rep2%'
or D.slide_id like '%AtGen_6-8111_Woundingstress-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-8112_Woundingstress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-8124_Woundingstress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-8126_Woundingstress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-8211_Woundingstress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-8214_Woundingstress-Shoots-1.0h_Rep2%'
```

```
or D.slide_id like '%AtGen_6-8224_Woundingstress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-8225_Woundingstress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-8313_Woundingstress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-8314_Woundingstress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-8324_Woundingstress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-8325_Woundingstress-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-8411_Woundingstress-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-8412_Woundingstress-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-8423_Woundingstress-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-8424_Woundingstress-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-8511_Woundingstress-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-8512_Woundingstress-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-8524_Woundingstress-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-8525_Woundingstress-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-9111_Heatstress-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-9112_Heatstress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-9121_Heatstress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-9122_Heatstress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-9211_Heatstress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-9212_Heatstress-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-9221_Heatstress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-9222_Heatstress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-9311_Heatstress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-9312_Heatstress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-9321_Heatstress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-9322_Heatstress-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-9411_Heatstress(3h)+3hrecovery-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-9412_Heatstress(3h)+3hrecovery-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-9421_Heatstress(3h)+3hrecovery-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-9422_Heatstress(3h)+3hrecovery-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-9511_Heatstress(3h)+9hrecovery-Shoots-12.0h_Rep1%'
```

```
or D.slide_id like '%AtGen_6-9512_Heatstress(3h)+9hrecovery-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-9521_Heatstress(3h)+9hrecovery-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-9522_Heatstress(3h)+9hrecovery-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-9611_Heatstress(3h)+21hrecovery-Shoots-24.0h_Rep1%'
or D.slide_id like '%AtGen_6-9612_Heatstress(3h)+21hrecovery-Shoots-24.0h_Rep2%'
or D.slide_id like '%AtGen_6-9621_Heatstress(3h)+21hrecovery-Roots-24.0h_Rep1%'
or D.slide_id like '%AtGen_6-9622_Heatstress(3h)+21hrecovery-Roots-24.0h_Rep2%'
or D.slide_id like '%AtGen_6-1111_Cold(4?C)-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-1112_Cold(4?C)-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-1121_Cold(4?C)-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-1122_Cold(4?C)-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-1211_Cold(4?C)-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-1212_Cold(4?C)-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-1221_Cold(4?C)-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-1222_Cold(4?C)-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-1311_Cold(4?C)-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-1312_Cold(4?C)-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-1321_Cold(4?C)-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-1322_Cold(4?C)-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-1411_Cold(4?C)-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-1412_Cold(4?C)-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-1421_Cold(4?C)-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-1422_Cold(4?C)-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-1511_Cold(4?C)-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-1512_Cold(4?C)-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-1521_Cold(4?C)-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-1522_Cold(4?C)-Roots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-1611_Cold(4?C)-Shoots-24.0h_Rep1%'
or D.slide_id like '%AtGen_6-1612_Cold(4?C)-Shoots-24.0h_Rep2%'
or D.slide_id like '%AtGen_6-1621_Cold(4?C)-Roots-24.0h_Rep1%'
or D.slide_id like '%AtGen_6-1622_Cold(4?C)-Roots-24.0h_Rep2%'
```

```
or D.slide_id like '%AtGen_6-2111_Osmoticstress-Shoots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-2112_Osmoticstress-Shoots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-2121_Osmoticstress-Roots-0.5h_Rep1%'
or D.slide_id like '%AtGen_6-2122_Osmoticstress-Roots-0.5h_Rep2%'
or D.slide_id like '%AtGen_6-2211_Osmoticstress-Shoots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-2212_Osmoticstress-Shoots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-2221_Osmoticstress-Roots-1.0h_Rep1%'
or D.slide_id like '%AtGen_6-2222_Osmoticstress-Roots-1.0h_Rep2%'
or D.slide_id like '%AtGen_6-2311_Osmoticstress-Shoots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-2312_Osmoticstress-Shoots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-2321_Osmoticstress-Roots-3.0h_Rep1%'
or D.slide_id like '%AtGen_6-2322_Osmoticstress-Roots-3.0h_Rep2%'
or D.slide_id like '%AtGen_6-2411_Osmoticstress-Shoots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-2412_Osmoticstress-Shoots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-2421_Osmoticstress-Roots-6.0h_Rep1%'
or D.slide_id like '%AtGen_6-2422_Osmoticstress-Roots-6.0h_Rep2%'
or D.slide_id like '%AtGen_6-2511_Osmoticstress-Shoots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-2512_Osmoticstress-Shoots-12.0h_Rep2%'
or D.slide_id like '%AtGen_6-2521_Osmoticstress-Roots-12.0h_Rep1%'
or D.slide_id like '%AtGen_6-2522_Osmoticstress-Roots-12.0h_Rep2%');
```

Appendix C

Perl Code to Retrieve *Entrez* Data Using E-Utilities

C.1 Perl Code: createGeneDB.pl

```
#!/usr/bin/perl
use LWP::Simple;
use NCBI_PowerScripting;
use strict;
use DBI;
##############################################################################
# Filename: createGeneDB.pl                                                  #
# Date Completed: November 3, 2004                                           #
# Place: Virginia tech                                                       #
#                                                                            #
# Function: Based on the organism name it downloads all the genes            #
# for that organism from NCBI's gene database from Entrez                    #
#                                                                            #
# Output: RDBMS Gene with the following fields                               #
# Protein_ID(gi:30686768)|Protein_Acc(NP_849441)|                            #
# Protein_Ver(NP_849441.1)|Protein_Annot(Superoxide dismutase)|              #
# Protein_Tax(3701)|Protein_Len(212)|Protein_CDD(567:890)|                   #
# Protein_Locus(At4g25700)|Protein_GeneID(828613)|Protein_Asm(F6N7.13)       #
# Protein_AA(MMATSG...)|Protein_Pubmed(56789)                                #
#                                                                            #
# Note that some of these fields may be absent based on their presence       #
# in the gp record retrieved                                                 #
```

```
#                                                                              #
# What needs to be set:                                                        #
# $ORG: This should be set to the name of the organism whose genes you         #
# want to retrieve                                                             #
#                                                                              #
# $term: This should be set to the combination of attributes of search         #
# for eg: Here we search for Heat Shock Proteins of the HSP 70 family          #
# in Arabidopsis thaliana. So the value for term is:                           #
# $term = 'Heat+Shock+protein+70[KYWD]+AND+Arabidopsis+thaliana[ORGN]'          #
#                                                                              #
# Need More Help? Contact: apati@vt.edu                                        #
################################################################################
my $email = 'apati@vt.edu';
my $ORG = 'Arabidopsis+thaliana';
my $baseURL = 'http://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
my ($begin, $end, $difftime);
my $maxdelay=3;
my $term='Arabidopsis+thaliana[ORGN]';
my $db_host='expresso';
my $user='apati';
my $passwd='**********';
my $dbname='pops';
my $conn = DBI->connect("dbi:Pg:dbname=${dbname};
           host=${db_host}", $user, $passwd, {RaiseError => 1});
my $insert_cmd = '';

#######################################################################
# STEP 1                                                              #
# Retrieve IDs of all Arabidopsis thaliana genes and                  #
# post them on the history server                                     #
#######################################################################
my $URL=$baseURL."esearch.fcgi?usehistory=y&db=gene&term=$term
                  &retmax=20&email=$email";
my $results=get($URL);
print "\nAll gene IDs retrieved and posted on history server.\n";
my $nrecords=0;
my $qkey='';
my $webenv='';
$results=~/<Count>(\d+)<\/Count>.*<QueryKey>(\d+).*<WebEnv>(.+)<\/WebEnv/s;
($nrecords,$qkey,$webenv)=($1,$2,$3);
print "\n$nrecords records to be retrieved\n";

#######################################################################
# STEP 2                                                              #
# Use gene IDs stored in history above to do an efetch with           #
# rettype: gb and retmode: xml for all of these ids and store them    #
```

```
# on history. Then download them in batches of 500 and parse out        #
# gene information to store in file.                                     #
##########################################################################
open(OUTFILE, ">gene.txt");
my $retstart=0;
my $retmax=500;
$begin=time;
my $basetime=time;
my ($geneID, $geneLocus, $geneAsm, $geneChr, $geneNuc, $geneAnnot,
    $geneTax, $geneGO, $geneStrand, $geneStart, $geneEnd,
    $genePub, $geneProt);
for($retstart=0;$retstart<$nrecords;$retstart+=$retmax){
  $end=time;
  $difftime=$end-$begin;
  if(3>$difftime) {print "\nSleeping...\n"; sleep(3-$difftime);}
  my $per=($retstart/$nrecords)*100;
  my $sofar=time-$basetime;
  print STDERR "\nElapsed time: $sofar seconds.\t$per% of records retrieved.\n";

  my $URL=$baseURL."efetch.fcgi?usehistory=y&db=gene&
          retstart=$retstart&retmax=$retmax&rettype=gb&
          retmode=xml&WebEnv=$webenv&query_key=$qkey";
  print "\n\n$URL\n";
  $begin=time;
  $results=get($URL);

  my @records=split(/^/,$results);
  my ($getgene,$gettax,$getGO,$getP,$setPar,
      $endPar,$twolines,$annotset,$getchr);
  $getgene=0;$gettax=0;$getGO=0;$getP=0;$setPar=0;
   $endPar=1;$twolines=0;$annotset=0;$getchr=0;

  foreach my $line(@records) {
    chomp($line);

    $getgene=1 if($line=~/<Entrezgene>/);

    if($line=~/<\/Entrezgene>/) {
      $insert_cmd="insert into gene values ('$geneID','$geneLocus',
                   '$geneNuc','$geneChr','$geneAsm','$geneAnnot',
                   '$geneTax','$geneGO','$geneStrand','$geneStart',
                   '$geneEnd','$genePub')";
      my $sth=$conn->do($insert_cmd);

      $getgene=0;
      $genePub='';
```

```
  $geneProt='';
  $annotset=0;
  $twolines=0;
  $setPar=0;
  $endPar=1;
  $getchr=0;
  $geneNuc='';
}

if($getgene==1) {
  #Get Gene ID
  if($line=~/<Gene-track_geneid>(\S+)<\/Gene-track_geneid>/) {
    if($1 ne '') {
      $geneID=$1;
    } else {
      $geneID='';
    }
  }
  #Get Gene Taxonomy
  $gettax=1 if($line=~/<Dbtag_db>taxon<\/Dbtag_db>/);
  if($gettax==1) {
    if($line=~/<Object-id_id>(\S+)<\/Object-id_id>/) {
      if($1 ne '') {
        $geneTax=$1;
      } else {
        $geneTax='';
      }
      $gettax=0;
    }
  }
  #Get Chromosome Number
  $getchr=1 if($line=~/<SubSource_subtype value="chromosome">
               (\S+)<\/SubSource_subtype>/);

  if($getchr==1) {
    if($line=~/<SubSource_name>(\S+)<\/SubSource_name>/) {
      if($1 ne '') {
        $geneChr=$1;
      } else {
        $geneChr='';
      }
      $getchr=0;
    }
  }
  #Get Assembly ID
  if($line=~/<Gene-ref_syn_E>(\S+)<\/Gene-ref_syn_E>/) {
```

```
  if($1 ne '') {
    $geneAsm=$1;
  } else {
    $geneAsm='';
  }
}
#Get Locus ID
if($line=~/<Gene-ref_locus-tag>(\S+)<\/Gene-ref_locus-tag>/) {
  if($1 ne '') {
    $geneLocus=$1;
  } else {
    $geneLocus='';
  }
}
#Get Annotation
if(($line=~/<Prot-ref_name_E>(.+)<\/Prot-ref_name_E>/) && ($annotset==0)) {
  if($1 ne '') {
    $geneAnnot=$1;
  } else {
    $geneAnnot='';
  }
  $annotset=1;
}
if (($line=~/<Prot-ref_name_E>(.+)/) && ($annotset==0)) {
  if($1 ne '') {
    $geneAnnot=$1;
  } else {
    $geneAnnot='';
  }
  $twolines=1;
}
if(($line=~/(.+)<\/Prot-ref_name_E>/) && ($annotset==0) && ($twolines==1)) {
  if($1 ne '') {
    #print "\nInside here $geneAnnot\n";
    $geneAnnot=$geneAnnot.$1;
    $twolines=0;
    $annotset=1;
  }
}
#Get Nucleotide ID.....................................................
$setPar=1 if(($line=~/<Gene-commentary_type value="genomic">(\d+)
             <\/Gene-commentary_type>/) && ($endPar==1));

if ($setPar==1) {
  #Get Start Position of Gene on chromosome
  if($line=~/<Seq-interval_from>(\d+)/) {
```

```
    if($1 ne '') {
      $geneStart=$1;
    } else {
      $geneStart=0;
    }
  }
  #Get End Position of Gene on chromosome
  if($line=~/<Seq-interval_to>(\d+)/) {
    if($1 ne '') {
      $geneEnd=$1;
    } else {
      $geneEnd=0;
    }
  }
  #Get strand on which gene is located
  if($line=~/<Na-strand value="(\w+)"/) {
    if($1 ne '') {
      $geneStrand=$1;
    } else {
      $geneStrand=0;
    }
    if($geneStrand eq "minus") {$geneStrand=2;} else {$geneStrand=1;}
  }
  #Get Nucleotide ID
  if($line=~/<Seq-id_gi>(\S+)<\/Seq-id_gi>/) {
    if($1 ne '') {
      $geneNuc=$1 if($geneNuc eq '');
    } else {
      $geneNuc='';
    }
    $setPar=0;
    $endPar=0;
  }
}
#Get pubmedIDs
if($line=~/<PubMedId>(\S+)<\/PubMedId>/) {
  if($1 ne '') {
    $genePub=$genePub.":".$1 if($genePub ne '');
    $genePub=$1 if($genePub eq '');
  }
}
#Get GO ID
$getGO=1 if($line=~/<Dbtag_db>GO<\/Dbtag_db>/);
if($getGO==1) {
  if($line=~/<Object-id_id>(\S+)<\/Object-id_id>/) {
    if($1 ne '') {
```

```
            $geneGO=$1;
          } else {
            $geneEnd=0;
          }
          $getGO=0;
        }
      }
      #Get IDs of protein links
      $getP=1 if($line=~/<Xtra-Terms_tag>PROTEIN<\/Xtra-Terms_tag>/);
      if($getP==1) {
        if($line=~/<Xtra-Terms_value>(\S+)<\/Xtra-Terms_value>/) {
          if($1 ne '') {
            $geneProt=$geneProt.":".$1 if($geneProt ne '');
            $geneProt=$1 if($geneProt eq '');
          }
          $getP=0;
        }
      }
    }#end of if $getgene=1
  }#end of foreach my $line
}
close(OUTFILE);
$conn->disconnect if($conn);
```

C.2 Perl Code: createProteinDB.pl

```
#!/usr/bin/Perl
use LWP::Simple;
use NCBI_PowerScripting;
use strict;
use DBI;
##############################################################################
# Filename: createProteinDB.pl                                               #
# Date Completed: November 14, 2004                                          #
# Place: Virginia tech                                                       #
#                                                                            #
# Function: Based on the organism name it downloads all the proteins         #
# for that organism from NCBI's protein database from Entrez                 #
#                                                                            #
# Output: RDBMS Protein with the following fields                            #
# Protein_ID(gi:30686768)|Protein_Acc(NP_849441)|                            #
# Protein_Ver(NP_849441.1)|Protein_Annot(Superoxide dismutase)|              #
# Protein_Tax(3701)|Protein_Len(212)|Protein_CDD(567:890)|                   #
# Protein_Locus(At4g25700)|Protein_GeneID(828613)|Protein_Asm(F6N7.13)       #
# Protein_AA(MMATSG...)|Protein_Pubmed(56789)                                #
#                                                                            #
```

```
# Note that some of these fields may be absent based on their presence  #
# in the gp record retrieved                                            #
#                                                                       #
# What needs to be set:                                                 #
# $ORG: This should be set to the name of the organism whose genes you  #
# want to retrieve                                                      #
#                                                                       #
# $term: This should be set to the combination of attributes of search  #
# for eg: Here we search for Heat Shock Proteins of the HSP 70 family   #
# in Arabidopsis thaliana. So the value for term is:                    #
# $term = 'Heat+Shock+protein+70[KYWD]+AND+Arabidopsis+thaliana[ORGN]'  #
#                                                                       #
# Need More Help? Contact: apati@vt.edu                                 #
#########################################################################

my $email = 'apati@vt.edu';
my $ORG = 'Arabidopsis+thaliana';
my $baseURL = 'http://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
my $UpBases = 1000;
my ($begin, $end, $difftime);
my $maxdelay=3;
my $term='Arabidopsis+thaliana[ORGN]';
my $db_host='expresso';
my $user='apati';
my $passwd='**********';
my $dbname='pops';
my $conn = DBI->connect("dbi:Pg:dbname=${dbname};
                         host=${db_host}", $user, $passwd, {RaiseError => 1});
my $insert_cmd = '';
my $protCount=0;

#####################################################################
# STEP 1                                                            #
# Retrieve IDs of all Arabidopsis thaliana genes and                #
# post them on the history server                                   #
#####################################################################
my $URL=$baseURL."esearch.fcgi?usehistory=y&db=gene&term=
        $term&retmax=20&email=$email";
my $results=get($URL);
print "\nAll gene IDs retrieved and posted on history server.\n";
my $nrecords=0;
my $qkey='';
my $webenv='';
$results=~/<Count>(\d+)<\/Count>.*<QueryKey>(\d+).*<WebEnv>(.+)<\/WebEnv/s;
($nrecords,$qkey,$webenv)=($1,$2,$3);
print "\n$nrecords records to be retrieved from $webenv and $qkey\n";
```

```
my $retstart=0;
my $retmax=150;
$begin=time;
my $results1='';
my $results2='';
my $basetime=time;
my $protIDList='';
my %protRecord;
for($retstart=0;$retstart<$nrecords;$retstart+=$retmax){ #Loop1
  $protCount=0;
  $protIDList='';
  $end=time;
  $difftime=$end-$begin;
  if(3>$difftime) {print "\nSleeping...gene...\n"; sleep(3-$difftime);}
  my $per=($retstart/$nrecords)*100;
  my $sofar=time-$basetime;
  print STDERR "\nElapsed time: $sofar seconds.\t
                $per% of gene and corresponding protein records retrieved.\n";

  $URL=$baseURL."esummary.fcgi?db=gene&retstart=$retstart&
                 retmax=$retmax&retmode=xml&WebEnv=$webenv&
                 query_key=$qkey";
  $begin=time;
  $results1=get($URL);
  #Now the geneIDs have to be extracted from the summary and appended to the URL
  #for elinking it to protein and then parsed to get geneID-proteinID
  my @records=split(/^/,$results1);
  my $linkURL=$baseURL."elink.fcgi?&db=protein&dbfrom=gene&email=$email";
  my ($getSumm);
  $getSumm=0;

  foreach my $line(@records) {
    chomp($line);
    $getSumm=1 if($line=~/<DocSum>/);
    $getSumm=0 if($line=~/<\/DocSum>/);
    if($getSumm==1) {
      if($line=~/<Id>(\d+)<\/Id>/) {
        $linkURL=$linkURL."&id=$1";
      }

    }
  }
  #At the end of this loop the retmax ids have been appended to the linking URL.
  #We just have to check the timing and get the linkURL now

  $end=time;
```

```
$difftime=$end-$begin;
if(3>$difftime) {print "\nSleeping...protein.elink..\n"; sleep(3-$difftime);}
$begin=time;
$results2=get($linkURL);
my @linkRec=split(/^/,$results2);
my ($getRec,$getGeneID,$getProtID, $geneID,$protID);
$getRec=0;$getGeneID=0;$getProtID=0;$geneID=0;$protID=0;

open(LINKOUTFILE,">>gene2protein.txt");
my $iter=1;
#Parse out the geneIDs and corresponding linked ProteinIDs
foreach my $linkline(@linkRec) {
  chomp($linkline);
  $getRec=1 if($linkline=~/<LinkSet>/);
  $getRec=0 if($linkline=~/<\/LinkSet>/);
  if($getRec==1) {
    $getGeneID=1 if($linkline=~/<IdList>/);
    $getGeneID=0 if($linkline=~/<\/IdList>/);
    if($getGeneID==1) {#Read gene ID here
      if($linkline=~/<Id>(\d+)<\/Id>/) {
        $geneID=$1 ;
      }
    }
    $getProtID=1 if($linkline=~/<LinkSetDb>/);
    $getProtID=0 if($linkline=~/<\/LinkSetDb>/);
    if($getProtID==1) {
      if($linkline=~/<Id>(\d+)<\/Id>/) {
        $protID=$1 ;
        if($iter==1) {
          $protIDList=$protIDList.$protID;
          $iter=$iter+1;
        } else {
          $protIDList=$protIDList.",".$protID;
        }
        $protCount=$protCount+1;
        print LINKOUTFILE "$geneID\t$protID\n";
        $insert_cmd="insert into gene2protein values ('$geneID','$protID')";
        my $sth=$conn->do($insert_cmd);
      }
    }
  }
}
close(LINKOUTFILE);
my $Pretstart=0; my $Pretmax=500;
for($Pretstart=0;$Pretstart<$protCount;$Pretstart+=$Pretmax){#Loop2
  my $protfetchURL=$baseURL."efetch.fcgi?usehistory=y&db=protein&
```

```
                retstart=$Pretstart&retmax=$Pretmax&rettype=gp&
                retmode=xml&id=$protIDList&email=$email";
$end=time;
$difftime=$end-$begin;
if(3>$difftime) {print "\nSleeping...protein.efetch..\n"; sleep(3-$difftime);}
$begin=time;
my $results4=get($protfetchURL);
#Now parse efetch results to prepare protein records
my @ProtFetch=split(/^/,$results4);
my $Precstart; my $PgeneName=0; my $Plocus=0; my $PgeneID=0; my $PprevCDD='';
my ($protID, $protAcc, $protVer, $protAnnot, $protTax, $protLen,
    $protCDD, $protLocus, $protGeneID, $protGeneAsm, $protSeq, $protPub);
foreach my $Protline(@ProtFetch) { #Loop3
  chomp($Protline);
  $Precstart=1 if($Protline=~/<GBSeq>/);
  if($Protline=~/<\/GBSeq>/) {
    $Precstart=0;
    $PprevCDD='';
    if(exists($protRecord{$protID})) {
      ;
    } else {
      $protRecord{$protID}=1;
      my $Pinsert_cmd="insert into protein values ('$protID','$protAcc',
                       '$protVer','$protLen','$protTax','$protCDD','$protPub',
                       '$protAnnot','$protGeneID','$protLocus','$protSeq')";
      my $FLAG=1;
      my $sth = $conn->prepare($Pinsert_cmd) or $FLAG=0;
      if($FLAG==1) {
        my $res=$sth->execute;
      }

    }
    $protCDD='';$protAnnot='';$protAcc='';$protVer='';$protLen=0;
    $protID='';$protTax='';$protGeneAsm='';$protGeneID='';$protLocus='';
  }
  if($Precstart==1) {
    if($Protline=~/<GBSeq_primary-accession>(\S+)
       <\/GBSeq_primary-accession>/) {
      if ($1 ne '') {
        $protAcc=$1;
      }
      else {
        $protAcc='';
      }
    }
    if($Protline=~/<GBSeq_accession-version>(\S+)
```

```
   <\/GBSeq_accession-version>/) {
  if ($1 ne '') {
    $protVer=$1;
  }
  else {
    $protVer='';
  }
}
if($Protline=~/<GBSeq_length>(\d+)<\/GBSeq_length>/) {
  if ($1 ne '') {
    $protLen=$1;
  }
  else {
    $protLen=0;
  }
}
if($Protline=~/<GBSeqid>gi\|(\S+)<\/GBSeqid>/) {
  if ($1 ne '') {
    $protID=$1;
  }
  else {
    $protID='';
  }
}
if($Protline=~/ <GBQualifier_value>taxon:(\S+)<\/GBQualifier_value>/) {
  if ($1 ne '') {
    $protTax=$1;
  }
  else {
    $protTax='';
  }
}
if($Protline=~/<GBQualifier_value>CDD:(\S+)<\/GBQualifier_value>/) {
  if ($1 ne '') {
    if($protCDD eq '') {
      $protCDD=$1;
      $PprevCDD=$1;
    }
    else {
      if($PprevCDD ne $1) {
        $protCDD=$protCDD.":".$1;
        $PprevCDD=$1;
      }
    }
  }
  else {
```

```
    $protCDD='';
  }
}
if($Protline=~/<GBSeq_sequence>(\S+)<\/GBSeq_sequence>/) {
  if ($1 ne '') {
    $protSeq=$1;
  }
  else {
    $protSeq='';
  }
}
if($Protline=~/<GBReference_pubmed>(\S+)<\/GBReference_pubmed>/) {
  if ($1 ne '') {
    $protPub=$1;
  }
  else {
    $protPub='';
  }
}
if($Protline=~/<GBSeq_definition>(.+)<\/GBSeq_definition>/) {
  if ($1 ne '') {
    $protAnnot=$1;;
  }
  else {
    $protAnnot='';
  }
}
if($Protline=~/<GBQualifier_name>gene<\/GBQualifier_name>/) {
  $PgeneName=1;
}
if($PgeneName==1) {
  if($Protline=~/<GBQualifier_value>(\S+)<\/GBQualifier_value>/) {
    if ($1 ne '') {
      $protGeneAsm=$1;
    }
    else {
      $protGeneAsm='';
    }
    $PgeneName=0;
  }
}
elsif($Protline=~/<Seq-feat_comment>gene_id:(\S+)<\/Seq-feat_comment>/
      && $protGeneAsm eq '') {
  if ($1 ne '') {
    $protGeneAsm=$1;
  }
```

```
        else {
          $protGeneAsm='';
        }
      }
      elsif($Protline=~/<Prot-ref_name_E>protein (\S+)<\/Prot-ref_name_E>/
            && $protGeneAsm eq '') {
        if ($1 ne '') {
          $protGeneAsm=$1;
        }
        else {
          $protGeneAsm='';
        }
      }
      else {;}
      if($Protline=~/<GBQualifier_name>db_xref<\/GBQualifier_name>/) {
        $geneID=1;
      }
      if($geneID==1) {
        if($Protline=~/<GBQualifier_value>GeneID:
          (\S+)<\/GBQualifier_value>/) {
          if ($1 ne '') {
            $protGeneID=$1;
          }
          else {
            $protGeneID='';
          }
          $geneID=0;
        }
      }
      if($Protline=~/<GBQualifier_name>locus_tag<\/GBQualifier_name>/) {
        $Plocus=1;
      }
      if($Plocus==1) {
        if($Protline=~/<GBQualifier_value>(\S+)<\/GBQualifier_value>/) {
          if ($1 ne '') {
            $protLocus=$1;
          }
          else {
            $protLocus='';
          }
          $Plocus=0;
        }
      }
    }
  }#End of Loop3
}#End of loop2
```

```
}#End of loop1
$conn->disconnect if($conn);
```

C.3 Perl Code: ExtractATUpstream.pl

```
#!/usr/bin/perl
use LWP::Simple;
use NCBI_PowerScripting;
use strict;
use DBI;
################################################################################
# Filename: ExtractATUpstream.pl ...although it can extract any upstream#
# Date Completed: November 1, 2004                                             #
# Place: Virginia tech                                                         #
#                                                                              #
# Function: Based on the organism name and keywords in the                     #
# annotations of the genes you want to retrieve, this program retrieves #
# 1. List of gene IDs                                                          #
# 2. Table: geneID|AtNo|Annotation|Organism|Chromosome|AssemblyID              #
# 3. Table: geneID|proteinID|proteinAcc#|TaxID|Protein_Seqlen|ProteinSeq#
# 4. Table: geneID|nucleotideID of chromosome on which this gene exists|#
# coordinates of the gene on the chromosome|strand(+/-)|chromosome|            #
# upstream sequence                                                            #
#                                                                              #
# What needs to be set:                                                        #
# $ORG: This should be set to the name of the organism whose genes you  #
# want to retrieve                                                             #
#                                                                              #
# $term: This should be set to the combination of attributes of search  #
# for eg: Here we search for Heat Shock Proteins of the HSP 70 family   #
# in Arabidopsis thaliana. So the value for term is:                           #
# $term = 'Heat+Shock+protein+70[KYWD]+AND+Arabidopsis+thaliana[ORGN]'  #
#                                                                              #
# $UpBases: This should be set to the number of upstream bases you need #
#                                                                              #
# Need More Help? Contact: apati@vt.edu                                        #
################################################################################

my $email = 'apati@vt.edu';
my $ORG = 'Arabidopsis+thaliana';
my $baseURL = 'http://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
my $UpBases = 2000;
my ($begin, $end, $difftime);
my $db_host="expresso";
my $dbname="pops";
my $user="apati";
```

```
my $passwd="**********";
my $conn = DBI->connect("dbi:Pg:dbname=${dbname};host=${db_host}",
                        $user, $passwd, {RaiseError => 1});
my $insert_cmd = '';
######################################################################
# STEP 1                                                             #
# Retrieve IDs of all Arabidopsis thaliana genes                     #
######################################################################
my $db = 'gene';
#my $term = $ORG.'[orgn]';
my $term = 'Arabidopsis+thaliana[ORGN]';
print "\nPosting ID list on history server.\n";
my $URL = $baseURL."esearch.fcgi?usehistory=y&db=$db&term=$term
          &email=$email";
print "\n$URL";
my $results = get($URL);
print "\nFinished esearch. ID list posted on history server.\n";

#Parse results to get count, webenv and query key
my $nrecords=0;
my $qkey='';
my $webenv='';
$results=~/<Count>(\d+)<\/Count>.*<QueryKey>(\d+).*<WebEnv>(.+)<\/WebEnv/s;
($nrecords,$qkey,$webenv)=($1,$2,$3);

print "\n$nrecords\t$qkey\n$webenv\n";
my $basetime=time;
##############################################################################
# STEP 2                                                                     #
# Retrieve Upstream Regions of all the geneIDs stored in the webenv in step1#
# Parse gene efetch results to retrieve nucleotideID, startPosition,         #
# endposition, and strand. Then extract upstream subsequence of gene         #
# from nucleotide.                                                           #
# Nucleotide IDs in case of AT link to Arabidopsis Chromosomes               #
# Upstream regions are stored in the file GeneUpstream.txt along with other  #
# information like location of region on chromosome, strand, etc.            #
##############################################################################
my $upfile = 'GeneUpstream.txt';
open(OUTFILE, ">$upfile");

$begin=time;
my $retstart=0; my $retmax=500;

for($retstart=0;$retstart<$nrecords;$retstart+=$retmax) {

  $end=time;
```

```
$difftime=$end-$begin;
if(3>$difftime) {print "\nSleeping...gene..."; sleep(3-$difftime);}
my $per=($retstart/$nrecords)*100;
my $sofar=time-$basetime;
print STDERR "\nElapsed time: $sofar seconds.
              \t$per% of records fetched and parsed.\n";

my $fetchURL=$baseURL."efetch.fcgi?usehistory=y&
             db=gene&retstart=$retstart&retmax=$retmax&
             rettype=gb&retmode=xml&WebEnv=$webenv&query_key=$qkey";
$begin=time;
my $fetchRes = get($fetchURL);

my @fetchRec = split(/^/,$fetchRes);
my ($nucID,$genStart,$genStop,$start,$stop,$strand,
    $getrecord,$getChrom,$chromo,$getNuc,$seq,$temp,
    $geneID,$upURL,$recordend,$printed,$sequence,$st);
$getrecord=0;$getChrom=0;$getNuc=0;$strand='';
$recordend=0;$printed=0;$sequence='';$st='';

foreach my $line(@fetchRec) {

  if($line=~/<Entrezgene>/) {
    $getrecord=1; $recordend=0;$printed=0;
  }
  if(($line=~/<Gene-commentary_type value=\"mRNA\">(\d+)
             <\/Gene-commentary_type>/)&&($recordend==0)) {
    print "\nRecord Ended";
    $getrecord=0;$recordend=1;
    $genStart=$start; $genStop=$stop;
    #Fix Upstream Coordinates
    if($strand == 1) {
      $temp=$start;
      $start=$temp-$UpBases;
      $stop=$temp-1;
    }
    else {
      $temp=$stop;
      $start=$temp+1;
      $stop=$temp+$UpBases;
    }

    #Now retrieve upstream regions
    $upURL = $baseURL."efetch.fcgi?db=nucleotide&id=$nucID&
             retmode=text&rettype=fasta&seq_start=$start&
             seq_stop=$stop&strand=$strand";
```

```
  $end=time;
  if(3>$difftime) {
    sleep(3-$difftime);
    print "\nSleeping...upstream..........\n";
  }
  $begin=time;

  $seq=get($upURL); chomp($seq);
  my @seqlines=split(/^/,$seq);
  foreach my $sline(@seqlines) {
    chomp($sline);
    if($sline=~/^>ref/) { next; }
    else { $sequence=$sequence.$sline; }
  }
  $insert_cmd="insert into upstream values ('$geneID','$nucID',
               '$chromo','$st','$genStart','$genStop','$sequence')";
  my $FLAG=1;
  my $sth = $conn->prepare($insert_cmd) or $FLAG=0;
  if($FLAG==1) {
    my $res=$sth->execute;
  }
  $geneID='';$chromo='';$strand='';$start='';
  $stop='';$seq='';$nucID='';$sequence='';$st='';
  $printed=1;
}
if(($line=~/<\/Entrezgene>/)&&($printed==0)) {
  print "\nRecord Ended";
  $getrecord=0;$recordend=1;
  $genStart=$start; $genStop=$stop;
  #Fix Upstream Coordinates
  if($strand==1) {
    $temp=$start;
    $start=$temp-$UpBases;
    $stop=$temp-1;
  }
  else {
    $temp=$stop;
    $start=$temp+1;
    $stop=$temp+1000;
  }

  #Now retrieve upstream regions
  $upURL = $baseURL."efetch.fcgi?db=nucleotide&id=
           $nucID&retmode=text&rettype=fasta&seq_start=$start&
           seq_stop=$stop&strand=$strand";
  $end=time;
```

```
  if(3>$difftime) {
    sleep(3-$difftime);
    print "\nSleeping...before upstream fetch..........\n";
  }
  $begin=time;

  $seq=get($upURL);chomp($seq);
  my @seqlines=split(/^/,$seq);
  foreach my $sline(@seqlines) {
    chomp($sline);
    if($sline=~/^>ref/) { next; }
    else { $sequence=$sequence.$sline; }
  }
  $insert_cmd="insert into upstream
               values ('$geneID','$nucID','$chromo',
               '$st','$genStart','$genStop','$sequence')";
  my $FLAG=1;
  my $sth = $conn->prepare($insert_cmd) or $FLAG=0;
  if($FLAG==1) {
    my $res=$sth->execute;
  }
  $geneID='';$chromo='';$strand='';$start='';$stop='';$seq='';
  $nucID='';$sequence='';$st=''; $printed=1;
}
if($getrecord==1) {
  #Parse out geneID
  if($line=~/<Gene-track_geneid>(\d+)<\/Gene-track_geneid>/) {
    $geneID=$1;
  }
  #Parse out chromosome number
  $getChrom=1 if($line=~/<SubSource_subtype value=\"chromosome\">
                       (\d+)<\/SubSource_subtype>/);
  if($getChrom==1) {
    if($line=~/<SubSource_name>(\d+)<\/SubSource_name>/){
      $chromo=$1;
      $getChrom=0;
    }
  }
  #Parse out nucleotide id and subsequent values
  $getNuc=1 if($line=~/<Gene-commentary_type value=
                      \"genomic\">(\d+)<\/Gene-commentary_type>/);
  if($getNuc==1) {
    if($line=~/<Gene-commentary_accession>(\S+)<
               \/Gene-commentary_accession>/) {
      $nucID=$1;
      $getNuc=0;
```

```
        }
      }
      if($line=~/<Seq-interval_from>(\d+)/) {
        $start=$1;
      }
      if($line=~/<Seq-interval_to>(\d+)/) {
        $stop=$1;
      }
      if($line=~/<Na-strand value="(\w+)"/) {
        $strand=$1;
        if($strand eq "minus") {$strand=2;$st='-';} else {$strand=1;$st='+';}
      }
    }#end of if $getrecord=1
  }#end of foreach
}#end of for
close(OUTFILE);
$conn->disconnect if($conn);
```

Appendix D

Shell Scrits and Perl Code from the XcisClique Pipeline

D.1 Perl Code: GetPromoter.pl

```
#!/usr/bin/perl -w

##############################################################################
# Syntax: GetPromoter.pl <input file with AT numbers> <output file>     #
# Function: Connects to 'pops' database on expresso.cs.vt.edu           #
#           and retrieves upstream regions of the genes whose AGIs      #
#           are in the input file with AT numbers, one on each line     #
# Output: Upstream regions, AGI and strand on one line, upstream        #
#         sequence on next line                                         #
# Email: apati@vt.edu                                                   #
##############################################################################

use DBI;

$conn = DBI->connect("dbi:Pg:dbname=pops host=expresso.cs.vt.edu")
        or die $conn->errorMessage;

my $USAGE = "$0 <File with At Nums> <Output file>";
if(!@ARGV) {
        print "\n$USAGE\n";
        exit;
}

open(INFILE,"$ARGV[0]") or die "GetPromoter.pl:
     Couldn't open Input File $ARGV[0] for reading";
open(OUTFILE,">$ARGV[1]") or die "GetPromoter.pl:
     Couldn't open Output File $ARGV[1] for writing";
```

```
while ($line = <INFILE>){

  $line =~/([Aa][Tt][1-5][Gg][0-9]*)/;

  $query  = "SELECT  gene.gene_locus, upstream.strand, upstream.upstream_seq ";
  $query .= "FROM gene, upstream ";
  $query .= "WHERE gene.gene_id=upstream.gene_id AND gene.gene_locus = '$1'";

  $sth = $conn->prepare($query) or die $conn->errorMessage;
  $result = $conn->selectall_arrayref($sth);

  if (@$result) {
    $row = shift(@$result);

    $locus   = @$row[0];  # gene_locus
    $strand = @$row[1]; #strand
    $seq     = @$row[2];  # upstream_seq

    print OUTFILE ">$locus|$strand\n";
    print OUTFILE "$seq\n";
  }
}
close(INFILE);
close(OUTFILE);
```

D.2 Perl Code: SelectSequences.pl

```
#!/usr/bin/perl

##############################################################################
# Syntax: SelectSequences.pl <File1-AtNums> <File2-Seqfile> <Outfile>   #
#          <flag>                                                       #
# Function: If flag==1, extract all sequences from File2, whose AtNums  #
#     are in File1                                                       #
#           If flag==0, extract all seq from File2 whose AtNums are NOT #
#           in File1                                                     #
# Email: apati@vt.edu                                                  #
##############################################################################
use strict;
use Utilities;
my $USAGE = "$0 <File with AtNums> <File with sequences superset>
             <Output File> <flag>";

if(!@ARGV) {
```

```
  print "\n$USAGE\n";
  exit;
}

open(ATFILE, $ARGV[0]) or die "SelectSequences.pl:
                        Cannot open input file $ARGV[0] for reading";
open(SEQFILE, $ARGV[1]) or die "SelectSequences.pl:
                         Cannot open input file with full genome
                         upstream regions $ARGV[1]";
open(OUTFILE, ">$ARGV[2]") or die "SelectSequences.pl:
                            Cannot open output file $ARGV[2]";
(($ARGV[3] == 0) || ($ARGV[3] == 1)) or die "SelectSequences.pl:
                                      Flag must be either 0 or 1";
my @AtNums = <ATFILE>;
my @Seq = <SEQFILE>;
my $flag=1; my $num='';
my $header='';

foreach my $seq(@Seq) {
  chomp($seq);
  if(!($seq =~ /^>/)) {
    if($flag==$ARGV[3]) {
      print OUTFILE ">$num\n$seq\n";
    }
  }
  $flag=0;
  if($seq =~ /^>/) {
    $header=$seq;
    $seq =~ /([Aa][Tt][1-5][Gg][0-9]*)/;
    $seq=$1;
    #print "\n$header***$seq";
    foreach my $line(@AtNums){
      chomp($line);
      $line=Utilities::trimwhitespace($line);
      if($line eq $seq) {
        #print "Seqid=$seq, Atnum=$line\n";
        $flag=1;
        next;
      } else {
        $num=$seq;
      }
    }
  }
}

close(ATFILE);
```

```
close(SEQFILE);
close(OUTFILE);
```

D.3 Perl Code: GetXUpstream.pl

```
#!/usr/bin/perl

use strict;

##############################################################################
# Syntax: GetXUpstream.pl <Input Fasta File> <Output File>              #
#         <Number of Upstream bases to extract X>                       #
# Function: Extracts X basepairs from the input fasta file sequences    #
#           and output sequences of length X to output file             #
# Output: Same format as input file but sequences of length X           #
# Email: apati@vt.edu                                                   #
##############################################################################

my $USAGE = "$0 <Input Fasta File> <Output File>
             <Number of Upstream bases to extract X-between 1 and 2000>";
if(!@ARGV) {
        print "\n$USAGE\n";
        exit;
}
open(INFILE,$ARGV[0]) or die "GetXUpstream.pl:
                      Couldn't open Input File $ARGV[0] for reading";
open(OUTFILE,">$ARGV[1]") or die "GetXUpstream.pl:
                           Couldn't open Output File $ARGV[1] for writing";
(($ARGV[2]>0) && ($ARGV[2]<=2000)) or die "GetXUpstream.pl:
                                    Number of bases should be between 1 and 2000";

my @input=<INFILE>;
foreach my $line(@input) {
  chomp($line);

  #check for FASTA header
  if ($line =~ /^>/) {
    print OUTFILE "$line\n";
  }
  else {
    my $seq500=substr($line,-$ARGV[2]);
    print OUTFILE "$seq500\n";
  }
}
close(INFILE);
```

```
close(OUTFILE);
```

D.4 Perl Code: FindMotifs.pl

```
#!/usr/bin/perl -w
use strict;
##############################################################################
# Syntax: FindMotifs.pl <File1 containing Motifs> <File2 containing     #
#         upstream sequences to be scanned> <Ouput File>                 #
# Format: File1: <motifName><tab><MotifSequence><newline>                #
#     File2: <>AtNum|strand><newline><sequence><newline>                 #
# Function: Looks for motifs from File1 in sequences in File2            #
#           by exact matching and outputs matches and positions          #
# Last Updated: April 15, 2005                                           #
# Email: apati@vt.edu                                                    #
##############################################################################
my $USAGE = "\n$0 <file with motifs> <file with upstream sequences>
                  <Output File>\n";

unless(@ARGV) {
  print $USAGE;
  exit;
}

open(MOTIFFILE, $ARGV[0]) or die "FindMotifs.pl:
  Cannot open motif file file $ARGV[0]\n";
open(DNAFILE, $ARGV[1]) or die "FindMotifs.pl:
  Cannot open file with upstream sequences $ARGV[1]\n";
open(RESULTFILE, ">$ARGV[2]") or die "FindMotifs.pl:
  Cannot open file $ARGV[2] for writing\n";

my @motifs = <MOTIFFILE>;
my @dna = <DNAFILE>;

print "\nFindMotifs.pl: The following motifs will be
        searched for in the DNA sequences:\n";
print "@motifs";

my ($index,$header);
foreach my $seq (@dna) {
        my %superhash;
  my %starthash; my %endhash; my %tatastart; my %tataend;
  my %matchhash; my %mothash; my %strandhash; my $print2file=0;
  my @fields;
  my ($TATAs, $TATAe, $reverseFlag); $TATAs=0; $TATAe=0; $reverseFlag=0;
```

```
$index=0;
#discard blank line
if ($seq =~ /^\s*$/) {
  next;

#discard comment line
} elsif ($seq =~ /^\s*#/) {
  next;

#Keep FASTA header
} elsif ($seq =~ /^>/) {
  $header=$seq;
  chomp $header;
  #print "\nHeader: $header";
  #replace all tabs and pipes in the header with spaces
  $header =~ s/>//g;

#else look for motifs in sequence
} else {
  $print2file=1;
  my $seqlen = length ($seq);
  foreach my $line (@motifs) {
    my $TATAcert=0;
    chomp $line;
    @fields = split(/\t/, $line);
    my $motlen = length($fields[1]);
    my $motifBackup = $fields[1];
    my $revMot='';

    #Check if motif is a regular expression
    #print "\nLooking for \"$fields[0]\t$fields[1]\"";
    if (($fields[1] =~ /(\\|\[|\{|\^|\$|\*|\+|\?|\.)/)||
        ($fields[0] =~ /TATABOX/)) {
      $reverseFlag=0;
    } else {
      $reverseFlag=1;
    }
    if ($reverseFlag==1) {
      $revMot = RevComp($fields[1]);
      #print "\nReverse Motif: $revMot";
    }
    $fields[1] = AlterMotif($fields[1]);

    #Search for motifs on sense strand
    while ( $seq =~ m/$fields[1]/gi ) {
      my $endpos = -($seqlen-pos($seq)+1);
```

```
  my $startpos = 0;
  my $match = $&;
  $motlen=length($match);
  $startpos = $endpos-$motlen+1;

  #Fix TATABOX positions
  if ($fields[0]=~m/TATABOX/i) {
    if($TATAs==0 && $TATAe==0) {
      $TATAs=$startpos; $TATAe=$endpos;
    } else {
      if((abs($TATAs)>=abs($startpos)) && (abs($TATAe)>=abs($endpos))) {
        $TATAe=$endpos; $TATAs=$startpos;
      }
    }
  }
  if((!($fields[0]=~m/TATABOX/i)) && (abs($TATAs)<abs($endpos))) {
    $TATAcert=1;
  } else {
    ;
  }
  if($TATAcert==1 &&
    (!exists($superhash{$fields[0],$match,$startpos,$endpos}))) {
  #if($TATAcert==1) {
    #Incorporating index and hash
    $superhash{$fields[0],$match,$startpos,$endpos}=1;
    $index=$index+1;
    $matchhash{$index}=$match;
    $starthash{$index}=$startpos;
    $endhash{$index}=$endpos;
    $mothash{$index}=$fields[0];
    $tatastart{$index}=$TATAs; $tataend{$index}=$TATAe;
    $strandhash{$index}='+';
    print RESULTFILE "$header\t$fields[0]\t$match\t
                      $startpos\t$endpos\t+\t$TATAs\t$TATAe\n";
    $TATAcert=0;
  }
}
#Search for motifs on anti-sense strand
if($reverseFlag==1) {
  while ( $seq =~ m/$revMot/gi ) {
    my $endpos = -($seqlen-pos($seq)+1);
    my $startpos = 0;
    my $match = $&;
    $motlen=length($match);
    $startpos = $endpos-$motlen+1;
```

```
          if((!($fields[0]=~m/TATABOX/i)) && (abs($TATAs)<abs($endpos))) {
            $TATAcert=1;
          } else {
            ;
          }
          #Check that the occurence of this motif on the forward
          #strand in the exact
          #same position has not been counted
          my $k=0;my $dupflag=0;
          for ($k=1; $k<=$index; $k++) {
            if(($fields[0] eq $mothash{$k}) &&
               ($starthash{$k}==$startpos) && ($endhash{$k}==$endpos)) {
              $dupflag=1;
              last;
            }
          }
          #print "Name: $fields[0] Seq: $fields[1] Dupflag: $dupflag\n";
          if(($TATAcert==1) && ($dupflag==0)
              &&(!exists($superhash{$fields[0],$match,$startpos,$endpos}))) {
            #Incorporating index and hash
            $superhash{$fields[0],$match,$startpos,$endpos}=1;
            $index=$index+1;
            $matchhash{$index}=$match;
            $starthash{$index}=$startpos; $endhash{$index}=$endpos;
            $mothash{$index}=$fields[0];
            $tatastart{$index}=$TATAs; $tataend{$index}=$TATAe;
            $strandhash{$index}='-';
            print RESULTFILE "$header\t$fields[0]\t$match\t
                              $startpos\t$endpos\t-\t$TATAs\t$TATAe\n";
            $TATAcert=0;
          }
        }
      } #end of if reverseFlag
    } #end of for-each motif
  } #end of else, finished looking for motifs
}
close MOTIFFILE;
close DNAFILE;
close RESULTFILE;
exit;
##############################################################################
## Subroutine for replacing ambiguous nucleic acid codes with regexp    ##
## M    A or C                                                          ##
## R    A or G                                                          ##
## W    A or T                                                          ##
## S    C or G                                                          ##
```

```
## Y    C or T                                                          ##
## K    G or T                                                          ##
## V    A or C or G                                                     ##
## H    A or C or T                                                     ##
## D    A or G or T                                                     ##
## B    C or G or T                                                     ##
## N    A or C or G or T                                                ##
##########################################################################
sub AlterMotif {
        my $element = $_[0];
  $element =~ s/m/(a|c)/gi;
        $element =~ s/r/(a|g)/gi;
  $element =~ s/w/(a|t)/gi;
  $element =~ s/s/(c|g)/gi;
  $element =~ s/y/(c|t)/gi;
  $element =~ s/k/(g|t)/gi;
  $element =~ s/v/(a|c|g)/gi;
  $element =~ s/h/(a|c|t)/gi;
  $element =~ s/d/(a|g|t)/gi;
  $element =~ s/b/(c|g|t)/gi;
  $element =~ s/n/(a|c|g|t)/gi;

        #print "the element now is: ", $element;
        return $element;
}

#####################################################################
# Subroutine for calculating reverse complement                     #
#####################################################################
sub RevComp {
  my $sek = $_[0];
  my $revstr = reverse $sek;

  $revstr =~ s/\)/x/gi;
  $revstr =~ s/\(/\)/gi;
  $revstr =~ s/x/\(/gi;
  $revstr =~ s/a/x/gi;
  $revstr =~ s/t/a/gi;
  $revstr =~ s/x/t/gi;
  $revstr =~ s/c/x/gi;
  $revstr =~ s/g/c/gi;
  $revstr =~ s/x/g/gi;
  $revstr =~ s/m/(t|g)/gi;
  $revstr =~ s/r/(t|c)/gi;
  $revstr =~ s/w/(t|a)/gi;
  $revstr =~ s/s/(g|c)/gi;
```

```
  $revstr =~ s/y/(g|a)/gi;
  $revstr =~ s/k/(c|a)/gi;
  $revstr =~ s/v/(t|g|c)/gi;
  $revstr =~ s/h/(a|g|t)/gi;
  $revstr =~ s/d/(a|c|t)/gi;
  $revstr =~ s/b/(c|g|a)/gi;
  $revstr =~ s/n/(a|c|g|t)/gi;

  return $revstr;
}
```

D.5 Perl Code: Chi-Square.pl

```
#!/usr/bin/perl
use strict;

######################################################################
# Chi-Square.pl:  Builds contingency table.                        ##
#       Computes Chi-Square score                                  ##
# Syntax:  $0 <# in expt> <#-control> <countfile> <outfile>        ##
# email: apati@vt.edu                                              ##
# Date:  Feb 16, 2005                                              ##
######################################################################

my $USAGE = "$0 <# in expt> <#-control> <countfile> <outfile>";

if(!@ARGV) {
  print "$USAGE\n";
  exit;
}

my $expt_num=$ARGV[0];
my $control_num=$ARGV[1];

unless ( open(CNTFILE, $ARGV[2]) ) {
  print "\nChi-Square.pl: Cannot open file $ARGV[2]\n\n";
  exit;
}
unless ( open(OUTFILE, ">$ARGV[3]") ) {
  print "\nChi-Square.pl: Cannot open file $ARGV[3]\n\n";
  exit;
}

my @inarr=<CNTFILE>;
```

```
foreach my $line(@inarr) {
  chomp($line);
  my @fields=split(/\t/,$line);
  my $name=$fields[0]; my $a=$fields[1]; my $b=$fields[2];

  my ($o11,$o12,$o21,$o22,$t1y,$t2y,$tx1,$tx2,$tot);
  my ($e11,$e12,$e21,$e22);
  my ($s11,$s12,$s21,$s22);


  $o11=$a; $o12=$expt_num-$o11;
  $o21=$b-$a; $o22=$control_num-$b-$o12;

  $t1y=$o11+$o12; $t2y=$o21+$o22;
  $tx1=$o11+$o21; $tx2=$o12+$o22;
  $tot=$t1y+$t2y;

  #print "\n$t1y**$t2y**$tx1**$tx2\n";

  $e11=($t1y*$tx1)/$tot; $e12=($t1y*$tx2)/$tot;
  $e21=($t2y*$tx1)/$tot; $e22=($t2y*$tx2)/$tot;

  my $chisq=0;
  if($e11==0 && $e21==0) {
    $chisq=0;
    print OUTFILE "$name\t$chisq\n";
  } else {
    $s11=(($e11-$o11)**2)/$e11;
    $s12=(($e12-$o12)**2)/$e12;
    $s21=(($e21-$o21)**2)/$e21;
    $s22=(($e22-$o22)**2)/$e22;

    $chisq=$s11+$s12+$s21+$s22;
    print OUTFILE "$name\t$chisq\n";
  }
}
close(CNTFILE);
close(OUTFILE);
```

D.6 Shell Script: CountMotifs.sh

```
#!/bin/bash

#############################################################################
# CountMotifs.sh:   Counts the number of each kind of pattern           #
#                   occuring in arg1 in file in arg2                    #
```

```
# Syntax:  $0 <file-with-pattern-names> <output-of-FindMotif>           #
#          <output for genome> <output-file>                            #
# Email: apati@vt.edu                                                   #
#########################################################################
echo $1
echo $2
echo $3
for i in `cat $1`
do
#Print motif name
`echo -n -e "$i\t" >> $4`

#Print motif count in set of genes of interest
echo -n -e "`grep $i $2 | wc -l`" >> $4
echo -n -e "\t" >> $4

#Print motif count in background set of genes
echo -n -e "`grep $i $3 | wc -l`" >> $4

#Print newline
echo -n -e "\n" >> $4
done
```

D.7 Perl Code: MakeAprioriMatrix.pl

```
#!/usr/bin/perl -w
#####################################################################
# Input: 2-column gene-motif file                                  #
# Output: Apriori Binary Input Matrix                              #
# Syntax: $0 <2-column gene-motif file> <output file-name>      #
# Email: apati@vt.edu                                              #
# ##################################################################

use strict;

my $USAGE = "$0 <2-column gene-motif input file> <output file>";

if(!@ARGV) {
  print "$USAGE\n";
  exit;
}

open(INFILE,$ARGV[0]) or die "MakeAprioriMatrix.pl:
                              Cannot open input file $ARGV[0]";
open(OUTFILE,">$ARGV[1]") or die "MakeAprioriMatrix.pl:
                                  Cannot open Output file $ARGV[1]";
```

```
my (%motifhash, %genehash, %motifgene, %motind, %genind);
my ($mind, $gind);
$mind=0; $gind=0;
my @records=<INFILE>;

foreach my $line(@records) {
  chomp($line);
  my @fields = split(/\t/,$line);
  if(!exists $motifhash{$fields[1]}) {
    $mind++;
    $motifhash{$fields[1]}=$mind;
    $motind{$mind}=$fields[1];
  }
  if(!exists $genehash{$fields[0]}) {
    $gind++;
    $genehash{$fields[0]}=$gind;
    $genind{$gind}=$fields[0];
  }
  if(!exists $motifgene{$motifhash{$fields[1]}}{$genehash{$fields[0]}}) {
    $motifgene{$motifhash{$fields[1]}}{$genehash{$fields[0]}}=1;
  }
}
my $i=0; my $j=0;

print OUTFILE "X";
for ($i=1; $i<=$mind; $i++) {
  print OUTFILE "\t$motind{$i}";
}
print OUTFILE "\n";
$i=0;
for ($i=1; $i<=$gind; $i++) {
  print OUTFILE "$genind{$i}";
  for ($j=1; $j<=$mind; $j++) {
    if (exists $motifgene{$j}{$i}) {
      print OUTFILE "\t1";
    } else {
      print OUTFILE "\t0";
    }
  }
  print OUTFILE "\n";
}

close(INFILE);
close(OUTFILE);
```

D.8 Perl Code: Hypergeometric.pl

```
#!/usr/bin/perl -w

use strict;
use Shell;
use PDF;
use DBI;

##############################################################################
# Hypergeometric.pl:  Finds hypergeometric tail p-value for different #
#  itemsets from Greg's Apriori algorithm #
# Syntax: $0 <apriori input> <output file>  #
#  <sig-itemsets file> <#fam-genes>#
# Input: Output from Apriori Algorithm #
#  Output for entire genome from GetMotifCounts.pl #
# Output: Counts for different itemsets in our output & genome #
#  p-value for each of these counts #
# Dependency: Output from GetMotifCounts.pl sgould be called  #
#  genomemotifcount and placed in the same directory #
# Author: Amrita Pati, apati@vt.edu #
# Date: Feb 21, 2005 #
##############################################################################
# Database specific variables
my $db_host='expresso'; my $dbname='pops';
my $user='apati'; my $passwd='kilo-meter11';
my $conn=DBI->connect("dbi:Pg:dbname=${dbname};host=${db_host}",
                              $user, $passwd, {RaiseError => 1});
my $insert_cmd='';

my $USAGE="$0 <output-from-apriori>
 <output-file fore-back numbers for each itemset>
 <output file-significant itemsets> <# genes in family>
 <file with analysis details>";

if(!@ARGV) {
print "$USAGE\n";
exit;
}

open (INFILE,$ARGV[0]) or die "\nCouldn't open input file $ARGV[0]";
open (OUTFILE,">$ARGV[1]") or die "\nCouldn't open output file $ARGV[1]";
open (OUTFILE2,">$ARGV[2]") or die "\nCouldn't open output file $ARGV[2]";
open (ANAFILE,$ARGV[4]) or die "\nCouldn't open analysis file $ARGV[4]";

my $geneFamNum=$ARGV[3];
my @inset=<INFILE>;
```

```
my %mainhash;
my %probhash;
my %bchash;
#Get analysis ID
my @ana=<ANAFILE>; my $flin=$ana[0];
my @darr=split(/\t/,$flin);
my($ana_id, $fdr, $ana_name)=($darr[0], $darr[2], $darr[1]);

foreach my $patt(@inset) {
chomp($patt);
my $res=GenomeCount($patt);
$res=~/(\S+);(\d+);(\d+);(\d+)/;
my($itemset,$motifcount,$samplegenecnt,$genomegenecnt)=($1,$2,$3,$4);
$mainhash{$itemset}=$patt;

#Compute hypergeometric tail
my $n=28829; #Total genome gene population
my $c=$genomegenecnt; #Total number of special genes in genome
my $k=$geneFamNum; #Number of genes selected=number of genes in family
my $l=$samplegenecnt; #Number of special genes in selection
#print "\nn=$n;c=$c;k=$k;l=$l";
my $prob=hypergeometric_tail($n,$k,$c,$l);
#print "\nItemset probability: $prob";
#Populate hash
$probhash{$res}=$prob;

        #Populate bicliques table
        $itemset=~/itemset(\d+)/; my $apriori_id=$1;
my $bcvalue=$ana_id."_".$apriori_id."_".$motifcount."_".$samplegenecnt."_".$prob;
$bchash{$res}=$bcvalue;
#print "$patt\n**BCValue:$bcvalue\n";

#Check if the biclique already exists
my $cckq="SELECT * FROM bicliques WHERE ana_id=$ana_id AND apriori_id=$apriori_id
                  AND num_genes=$samplegenecnt AND num_motifs=$motifcount";
my $scck=$conn->prepare($cckq);
my $cckres=$conn->selectall_arrayref($scck);
if (!(@$cckres && (my $cckrow=shift(@$cckres)))) {
          $insert_cmd="INSERT into bicliques (ana_id,apriori_id,
                       num_genes,num_motifs,pval_hyp) values
                       ($ana_id,$apriori_id,$samplegenecnt,$motifcount,$prob)";
  my $sth=$conn->do($insert_cmd);
} else {
  print "BiClique already Exists\n";
}
```

```
#Get uid of biclique just inserted
my $bcid;
my $sq1="SELECT uid from bicliques WHERE ana_id='$ana_id'
                and apriori_id=$apriori_id ";
my $s1th=$conn->prepare($sq1) or die $conn->errorMessage;
my $result=$conn->selectall_arrayref($s1th);
while (@$result && (my $row=shift(@$result))) {
  $bcid=@$row[0];
}
$bchash{$res}=$bcid."_".$bchash{$res};

#Populate edges table
my @pieces=split(/\t/,$patt);
my @motifnames=splice @pieces, 2, $motifcount;
my @genenames=splice @pieces, 2, $samplegenecnt;
#print "@motifnames\n"; print "@genenames\n";
foreach my $gene(@genenames) {
  chomp($gene);
  #select gid
  my $gid;
  my $sq2="SELECT gene_id from gene where gene_locus ilike '\%$gene%'";
  my $s2th=$conn->prepare($sq2) or die $conn->errorMessage;
  $result=$conn->selectall_arrayref($s2th);
  while (@$result && (my $row=shift(@$result))) {
    $gid=@$row[0];
  }

  foreach my $motif(@motifnames) {
    chomp($motif);
    #select cisid
    my $cisid;
    my $sq3="SELECT uid from regulatory_motifs_new WHERE cisid ='$motif'";
    my $s3th=$conn->prepare($sq3) or die $conn->errorMessage;
    $result=$conn->selectall_arrayref($s3th);
    while (@$result && (my $row=shift(@$result))) {
      $cisid=@$row[0];
    }

    #Insert into edges table
    if($gid) {
      #Check that the edges don't already exist
      my $sq4="SELECT * from edges WHERE motif=$cisid AND gene=$gid
                       AND biclique=$bcid";
      my $s4th=$conn->prepare($sq4);
      my $s4res=$conn->selectall_arrayref($s4th);
      if (!(@$s4res && (my $s4row=shift(@$s4res)))) {
```

```
        $insert_cmd="INSERT into edges (motif,gene,biclique) values
                              ($cisid,$gid,$bcid)";
        $conn->do($insert_cmd);
              } else {
                print "Edge already exists\n";
      }
    }

  }#end of foreach motifs
}#end of foreach genes
}

#Sort hash numerically on values: arranges in ascending order
my($i); $i=1;
my %sechashkey; my %sechashval;

foreach(sort{$probhash{$a}<=>$probhash{$b}} keys %probhash) {
print "\nKey is $_ and value is $probhash{$_}";
$sechashkey{$i}=$_; $sechashval{$i}=$probhash{$_};

#Update ranks in bicliques table
my $ky=$_; chomp($ky);
my @vals=split(/_/,$bchash{$ky});
my $bc=$vals[0];
my $update_q="UPDATE bicliques set rank_fdr='$i' where uid='$bc'";
$conn->do($update_q);

$i=$i+1;
}
my $size=keys (%sechashkey);
print "\nSecondary Hash Size: $size";
#Do FDR correction and print
my $sig=$fdr;
for($i=0; $i<$size; $i++) {
print "\nOutside: i is $i***";
my $compare = (($i+1)*$sig)/$size;
print "Printing probability: $sechashval{$i} compared to $compare";
$sechashkey{$i}=~/(\S+);(\d+);(\d+);(\d+)/;
my($itemset,$motifcount,$samplegenecnt,$genomegenecnt)=($1,$2,$3,$4);
print OUTFILE "$itemset\t$motifcount\t$samplegenecnt\t
                      $genomegenecnt\t$sechashval{$i}\n";
if( ($sechashval{$i}) < ((($i+1)*$sig)/$size) ) {
print "***Inside: i is $i";
print OUTFILE2 "$mainhash{$itemset}\n";
}
}
```

```
$conn->disconnect if($conn);
close(INFILE); close(OUTFILE); close(OUTFILE2); close(ANAFILE);

##############################################################################
# GenomeCount: Subroutine, finds the number of occurrences of #
#  each itemset in the genome #
# Input: Itemset record #
# Returns: Count in genome #
##############################################################################

sub GenomeCount {

my $pattern=$_[0];

#Get the entire itemset data
my @pieces=split(/\t/,$pattern);
my $info=$pieces[0];
chomp($info);

#Parse out number of motifs and genes
$info=~/(\S+)_(\d+)_(\d+)/;
my($itemset,$nummotifs,$numgenes)=($1,$2,$3);
my @motifnames=splice @pieces, 2, $nummotifs;

my $motifs=join(',',@motifnames);
#print "\n$itemset....$nummotifs....$numgenes\nMotifs are $motifs";

cp('genomemotifcount','outfile');

foreach my $motif(@motifnames) {
qx(grep $motif outfile > tempfile);
cp('tempfile','outfile');
}
#count contains the number of genes in the genome that contain
        #the particular itemset
my $result= wc('-l', 'outfile');
$result=~/(\d+)(\s+)(\S+)/;
my($count,$filename)=($1,$2);
rm('outfile','tempfile');
$result=$itemset.";".$nummotifs.";".$numgenes.";".$count;
return $result;
}
```

D.9 Perl Code: Eval_Itemset.pl

```
##############################################################################
## Eval$\_$Itemsets-DB.pl: Finds how tightly correlated each ietmset is#
##                       with respect to expression data using SAV      #
## Syntax:  $0  <File with itemsets>                                      #
##              <File with p-values from hypergeometric>                  #
##              <Output File> <tissue> <array_set>                        #
## Email:  apati@vt.edu                                                   #
## Last Updated:    June 10, 2005                                         #
##############################################################################

#!/usr/bin/perl

use strict;
use Shell;
use DBI;
use Vector;
use Statistics::Distributions;

# Database specific variables
my $db_host='expresso'; my $dbname='pops'; my $user='apati';
my $passwd='kilo-meter11';
my $conn=DBI->connect("dbi:Pg:dbname=${dbname};host=${db_host}",
                      $user, $passwd, {RaiseError => 1});

my $USAGE="$0 <File with itemsets> <File with p-values from hypergeometric>
          <Output File> <Analysis file> <tissue> <treatment_set>\n";
if(!@ARGV) {
  print "$USAGE\n"; exit;
}
open(INFILE,$ARGV[0]) or die "Couldn't open input file $ARGV[0]\n";
open(INFILE1,$ARGV[1]) or die "Couldn't open input file $ARGV[1]\n";
open(OUTFILE,">$ARGV[2]") or die "Couldn't open output file $ARGV[2]\n";
open(INFILE3,$ARGV[3]) or die "Couldn't open input file $ARGV[3]\n";

my $tissue=$ARGV[4]; my $tset=$ARGV[5];
my $array_set=$ARGV[6];
my $ptissue="pval"."_".$tissue;
#Get analysis ID
my @ana=<INFILE3>; my $flin=$ana[0];
my @darr=split(/\t/,$flin);
my($ana_id, $fdr, $ana_name)=($darr[0], $darr[2], $darr[1]);
#Get array_set ID
=pod
my $sq="SELECT array_set FROM analysis WHERE uid=$ana_id";
my $someth=$conn->prepare($sq);
```

```
my $asres=$conn->selectall_arrayref($someth);
while(@$asres && (my $asrow=shift(@$asres))) {
  $array_set=@$asrow[0];
}
=cut

my %pvalhash;
while(<INFILE1>) {
  my $line=$_; chomp($line);
  my @arr=split(/\t/,$line);
  $pvalhash{$arr[0]}=$arr[4];
}

my %corrhash;
while(<INFILE>) {
  %corrhash=();
  my $patt=$_; chomp($patt);
  my @pieces=split(/\t/,$patt);
  my $info=$pieces[0]; $info=~/(\S+)_(\d+)_(\d+)/;
  my($itemset,$nummotifs,$numgenes)=($1,$2,$3);
  my @motifs=splice @pieces, 2, $nummotifs;
  my @genes=splice @pieces, 2, $numgenes;

  my $unknown=0;
  if($numgenes<=2) {
    next;
  } else {
    my $i=0; my $j;
    for($i=0; $i<@genes; $i++) {
      #print "AT1=@genes[$i]\n";
      for($j=$i+1; $j<@genes; $j++) {
        #print "AT2=@genes[$j]\n";
        if(exists($corrhash{$genes[$i],$genes[$j]})) {
          next;
        } else {
          my($pr,$pp,$sr,$sp,$kr,$kp);
          my $gene1=$genes[$i]; my $gene2=$genes[$j];
          my $selq="select pearson_r, pearson_significance, spearman_rho,
                    spearman_significance, kendall_tau, kendall_significance
                    from expr_correlations where ((gene1=(select gene_id from
                    gene where gene_locus~'$gene1') and gene2=(select gene_id
                    from gene where gene_locus~'$gene2')) or (gene1=(select
                    gene_id from gene where gene_locus~'$gene2') and gene2=
                    (select gene_id from gene where gene_locus~'$gene1')))
                    and array_set=$array_set and tissue='$tissue'";
          my $sth=$conn->prepare($selq);
```

```
        my $res=$conn->selectall_arrayref($sth);
if(!@$res) {
  print "Unknown Correlation\n"; $unknown++;
}
        while (@$res && (my $row=shift(@$res))) {
          $pr=@$row[0]; $pp=@$row[1]; $sr=@$row[2];
          $sp=@$row[3]; $kr=@$row[4]; $kp=@$row[5];
        }
#print "Pr=$pr,PP=$pp,Sr=$sr,Sp=$sp,Kr=$kr,Kp=$kp\n";
$corrhash{$genes[$i],$genes[$j]}=$pr."_".$pp."_".$sr."_".
                                          $sp."_".$kr."_".$kp;
      }#end of else
    }#end of inner for
  }#end of outer for
}#end of else

#while (my($key,$value)=each(%corrhash)) {
  #print "$key\t$value\n";
#}

#Correlations Retrieved. Now actually evaluate the itemset
my $count=0; my $corrVec='';
for(my $i=0; $i<@genes; $i++) {
  for(my $j=$i+1; $j<@genes; $j++) {
    $count++;
    my $vals;
    if(exists($corrhash{$genes[$i],$genes[$j]})) {
      $vals=$corrhash{$genes[$i],$genes[$j]};
    } elsif(exists($corrhash{$genes[$j],$genes[$i]})) {
      $vals=$corrhash{$genes[$j],$genes[$i]};
    } else {
      print "Unknown Correlation\n";
      $unknown++;
    }

    my @arr=split(/_/,$vals);
    my $sr=abs($arr[2]); my $sp=$arr[3];

    if($corrVec eq '') {
      $corrVec=$sr;
    } else {
      $corrVec=$corrVec."_".$sr;
    }
  }
}
print "$corrVec\n";
```

```
#my $mean=Vector::VectorMean($corrVec,'_');
my $S=Vector::VectorSum($corrVec,'_');
my $std=Vector::VectorSigma($corrVec,'_');
my ($mu,$sigma);

# Select values directly from table
my $selq="SELECT mean_sum_abs_corr FROM correlation_cv_dist
          WHERE treatment_set=$tset AND tissue='$tissue' AND
          samplesize=$numgenes AND num_iterations=100000;";
my $sth=$conn->prepare($selq);
my $res=$conn->selectall_arrayref($sth);
while(@$res && (my $row=shift(@$res))) {
  $mu=@$row[0];
}
#print "Mean=$mu;Unknown=$unknown\n";
my $numcomb=($numgenes*($numgenes-1))/2;
print "Num Combinations=$numcomb, Unknowns=$unknown, ";
$S=$S+($unknown*($mu/$numcomb));

print "S=$S\n";
my $Pval;
if($S>=$mu) {

  # Npw get S to nearest a.b0 or a.b5 form.
  my $Strunc=sprintf ("%.2f",$S);
  my $sq="select cdf from s_distribution where tissue='$tissue'
          and treatment_set=$tset and genesetsize=$numgenes and
          num_iterations=100000 and interval_upper_limit=
          (select interval_upper_limit from s_distribution
          where tissue='$tissue' and treatment_set=$tset and
          genesetsize=$numgenes and
          (abs($Strunc-interval_upper_limit))<0.025);";
  $sth=$conn->prepare($sq);
  $res=$conn->selectall_arrayref($sth);
  my $cdf;
  while(@$res && (my $row=shift(@$res))) {
    $cdf=@$row[0];
  }
  $Pval=1-$cdf;
} else {
  $Pval=1;
}
my $hyp=$pvalhash{$itemset};
my $FinalP=$Pval*$hyp;
print OUTFILE "$itemset\t$hyp\t$Pval\t$FinalP\n";
print "$itemset\t$hyp\t$S\t$Pval\t$FinalP\n";
```

```
  #Parse out apriori ID
  $itemset=~/itemset(\d+)/; my $bcid=$1;

  #And put p-values into the database
  my $upd_q="update bicliques set $ptissue=$Pval where
             ana_id=$ana_id and apriori_id=$bcid ";
  $conn->do($upd_q);
  print "Updated\n";
}#end of while INFILE

close(INFILE); close(INFILE1); close(INFILE3);
$conn->disconnect if($conn);
```

D.10 Shell Script to Run XcisClique: Compile.sh

```
#! /bin/sh
USAGE="Usage: ./compile.sh [OPTION]... [FILE/VALUE]...
Options:
    -c          0 indicates that user supplies all files, 1 indicates that
                program uses Arabidopsis upstream regions in the package
    -b          Number of upstream base-pairs to be analyzed
    -a          File containing Identifiers of genes to be analyzed
                One ID per line
    -u          File containing upstream sequences in fasta format
                Id and sequence on separate lines
                Id must be preceded by \">\"
    -m          File containing motifs in the format
                <motifname>\t<sequence>
                Must contain TATABOX motifs in the first few lines
    -r          Rootname of all result files. All result files will
                be preceded by this name
    -d          File with name and description of analysis
    -f          False Discovery Rate
    -z          Database to be populated? (Yes=1; No=0)
    --help      Print help and exit
This program was tested on linux-2.6.11-gentoo-r4.
Report bugs to <apati@vt.edu>";

if [ $# -eq 0 ]
then
  echo "$USAGE";
  exit 0;
fi
```

```
while [ $# -ge 1 ]; do
  case $1 in
    -c)shift; org=$1;
       ;;
    -b)shift; bps=$1;
       ;;
    -a)shift; AtFile=$1;
       ;;
    -u)shift; UpsFile=$1;
       ;;
    -m)shift; MotFile=$1;
       ;;
    -r)shift; rootname=$1;
       ;;
    -d)shift; anafile=$1;
       ;;
    -f)shift; fdr=$1;
       ;;
    -z)shift; database=$1;
       ;;
    --help)echo "$USAGE"; exit 1;
           ;;
    *)echo "Incorrect options specified"; echo "$USAGE"; exit 0;
      ;;
  esac
  shift
done

echo "Organism=$org; bps=$bps; AtFile=$AtFile; UpsFile=$UpsFile;
MotFile=$MotFile; Rootname=$rootname; FDR=$fdr; Anafile=$anafile;
Database=$database";
#Check if the necessary variables were populated
if [ -z $org ]
then
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
elif [ -z $bps ]; then
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
elif [ -z $AtFile ]; then
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
elif [ -z $rootname ]; then
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
elif [ -z $anafile ]; then
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
elif [ -z $fdr ]; then
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
elif [ -z $MotFile ]; then
```

```
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
elif [ -z $database ]; then
  echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
fi

#Check existence of AT number file
if [ ! -f $AtFile ]
then
  echo "File $AtFile doesn't exist"; exit 0;
fi

if [ $org -eq 1 ] #means organism is arabidopsis
then
  echo "The organism is Arabidopsis thaliana";
  backgroundNum=28829;
  foregroundNum=`wc -l $AtFile | awk -F' ' '{printf("%s",$1)}'`;
  echo "foregroundNum=$foregroundNum";

  if [ $database -eq 1 ]
  then
    echo "Populating Analysis Table";
    step0=`./scripts/PopulateAnalysisTable.pl $AtFile $anafile
          $fdr $foregroundNum`;
    echo "Analysis table populated";
  fi

  echo "Entering SelectSequences.pl"; echo "${rootname}_Upstream_2k";
  step1=`./scripts/SelectSequences.pl $AtFile ./data/AT_Genome_Upstream.fa
        results/${rootname}_Upstream_2k 1`;
  echo "Results file(s) from this step";
  echo "results/${rootname}_Upstream_2k";
  echo "Exiting SelectSequences.pl";

  #Do genomewide operations according to number of base pairs
  echo "Processing Genome Upstream";
  #g1=`scripts/GetXUpstream.pl ./data/AT_Genome_Upstream.fa
      ./data/AT_Genome_Upstream_$bps.fa $bps`;
  #g2=`scripts/FindMotifs.pl $MotFile ./data/AT_Genome_Upstream_$bps.fa
      ./data/AT_Genomewide_Motifs_$bps.txt`;
  g3=`scripts/GetMotifCounts.pl ./data/AT_Genomewide_Motifs_$bps.txt
     ./data/AT_Genewise_motif_cnts_$bps.txt ./data/AT_global_motif_cnts_$bps.txt`;
  echo "Ended Genome Upstream processing";

  #MotFile="data/AT_motifs";
  #MotFile="ATNUMLISTS/AT_motifs_ERD";
  BgOpMotFile="data/AT_Genewise_motif_cnts_$bps.txt";
```

```
  #Overwrite genomemotifcount
  val1=`cp data/AT_Genewise_motif_cnts_$bps.txt genomemotifcount`;
  #Fix Motif names file
  MotifNameFile="data/motif_names";

else #means user supplies his own upstream files
  echo "The organism is not Arabidopdis thaliana:
        Using user supplied motif and sequence files";

  #Check if the files supplied by the user exist
  if [ -z $UpsFile ]; then
    echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
  elif [ -z $MotFile ]; then
    echo "Please specify all necessary options"; echo "$USAGE"; exit 0;
  elif [ ! -f $UpsFile ]; then
    echo "File $UpsFile doesn't exist"; exit 0;
  elif [ ! -f $MotFile ]; then
    echo "File $MotFile doesn't exist"; exit 0;
  fi
  backgroundNum=`wc -l $UpsFile | awk -F' ' '{printf("%s",$1)}'`;
  foregroundNum=`wc -l $AtFile | awk -F' ' '{printf("%s",$1)}'`;
  echo "Entering SelectSequences.pl";
  step1=`./scripts/SelectSequences.pl $AtFile $UpsFile
        results/${rootname}_Upstream_2k 1`;
  echo "Results file(s) from this step";
  echo "results/${rootname}_Upstream_2k";
  echo "Exiting SelectSequences.pl";

  #Find genomewide motifs
  val2=`./scripts/FindMotifs.pl $MotFile $UpsFile
       results/${rootname}_background_output_motifs`;
  val3=`./scripts/GetMotifCounts.pl results/${rootname}_background_output_motifs
       results/${rootname}_background_Genewise_Cnts
       results/${rootname}_background_Global_Cnts`;
  BgOpMotFile="results/${rootname}_background_Genewise_Cnts";

  #Overwrite genomemotifcount
  val4=`cp results/${rootname}_background_Genewise_Cnts genomemotifcount`;

  #Make motif_names file
  val5=`cut -f 1 $MotFile > results/${rootname}_intfile`;
  val6=`grep -v TATABOX results/${rootname}_intfile >
        results/${rootname}_motif_names`;
  MotifNameFile="results/${rootname}_motif_names";
fi
```

```
echo "Entering GetXUpstream.pl";
step2=`./scripts/GetXUpstream.pl results/${rootname}_Upstream_2k
      results/${rootname}_Upstream_$bps $bps`;
echo "Results file(s) from this step";
echo "results/${rootname}_Upstream_$bps";
echo "Exiting GetXUpstream.pl";
echo "Entering FindMotifs.pl";
step3=`./scripts/FindMotifs.pl $MotFile results/${rootname}_Upstream_$bps
      results/${rootname}_Output`;
echo "Results file(s) from this step";
echo "results/${rootname}_Output";
echo "Exiting FindMotifs.pl";
echo "Entering GetMotifCounts.pl";
step4=`./scripts/GetMotifCounts.pl results/${rootname}_Output
      results/${rootname}_Genewise_Cnts results/${rootname}_Global_Cnts`;
echo "Results file(s) from this step";
echo "results/${rootname}_Genewise_Cnts";echo "results/${rootname}_Global_Cnts";

echo "Enter the appropriate number for the type of analysis you want:";
echo "1. Motifwise significance: Chi-Square Test";
echo "2. Determining significant motif combinations:
         Apriori + Hypergeometric Test";
read ans;
case $ans in
        1)echo "Entering CountMotifs.sh";
          echo "$BgOpMotFile, ${rootname}_Genewise_Cnts";
          step5=`./scripts/CountMotifs.sh $MotifNameFile
                results/${rootname}_Genewise_Cnts $BgOpMotFile
                results/${rootname}_ForeBack_Cnts`;
          echo "Results file(s) from this step";
          echo "results/${rootname}_ForeBack_Cnts";
          echo "Exiting CountMotifs.sh";
          echo "Entering Chi-Square.pl"
          step6=`./scripts/Chi-Square.pl $foregroundNum $backgroundNum
                results/${rootname}_ForeBack_Cnts
                results/${rootname}_ChiSquare_Scores`;
          echo "Results file(s) from this step";
          echo "results/${rootname}_ChiSquare_Scores";
          echo "Exiting Chi-Square.pl";
          ;;
        2)echo "Creating Apriori Matrix";
          step7=`cat results/${rootname}_Output |
                awk '{printf("%s\t%s\n",$1,$2)}'> results/${rootname}_GeneMotif`;
          step8=`./scripts/MakeAprioriMatrix.pl
                results/${rootname}_GeneMotif
                results/${rootname}_AprioriMatrix`;
```

```
      echo "Running Apriori";
      step9=`apriori-1.0/bin/apriori
            results/${rootname}_AprioriMatrix >
            results/${rootname}_Apriori_Itemsets`;
      echo "Results file(s) from this step";
      echo "results/${rootname}_Apriori_Itemsets";
      echo "Entering Hypergeometric.pl";
      if [ $database -eq 1 ]
      then
        step10=`scripts/Hypergeometric-DB.pl
results/${rootname}_Apriori_Itemsets
results/${rootname}_Itemsets_ForeBack_Cnts
results/${rootname}_Sig_Itemsets
$foregroundNum $anafile $fdr`;
        step12=`scripts/Eval_Itemsets-DB.pl
results/${rootname}_Apriori_Itemsets
results/${rootname}_Itemsets_ForeBack_Cnts
results/${rootname}_Itemsets_PValues_Roots
$anafile Roots $treatset`;
        step13=`scripts/Eval_Itemsets-DB.pl
results/${rootname}_Apriori_Itemsets
results/${rootname}_Itemsets_ForeBack_Cnts
results/${rootname}_Itemsets_PValues_Shoots
$anafile Shoots $treatset`;
      else
step10=`scripts/Hypergeometric.pl
results/${rootname}_Apriori_Itemsets
results/${rootname}_Itemsets_ForeBack_Cnts
results/${rootname}_Sig_Itemsets
$foregroundNum $fdr`;
        step12=`scripts/Eval_Itemsets.pl
results/${rootname}_Apriori_Itemsets
results/${rootname}_Itemsets_ForeBack_Cnts
results/${rootname}_Itemsets_PValues_Roots
Roots $treatset`;
        step13=`scripts/Eval_Itemsets.pl
results/${rootname}_Apriori_Itemsets
results/${rootname}_Itemsets_ForeBack_Cnts
results/${rootname}_Itemsets_PValues_Shoots
Shoots $treatset`;

      fi
      echo "Results file(s) from this step";
      echo "results/${rootname}_Itemsets_ForeBack_Cnts";
      echo "results/${rootname}_Sig_Itemsets";
      echo "Exiting Hypergeometric.pl";
```

```
          step11=`scripts/Sig2Output.pl
                 results/${rootname}_Sig_Itemsets
                 results/${rootname}_Output
                 results/${rootname}_Sig_Output`;
          echo "Significant results file: results/${rootname}_Sig_Output";
          echo "Entering Eval_Itemsets.pl";
          echo "Exiting Eval_Itemsets.pl";
          ;;
esac

exit 1;
```

Appendix E

Expression Data Analysis: Perl and MATLAB Code

E.1 Perl Code: MakeExpressionVectors.pl

```
#!/usr/bin/perl -w

use strict;
use Utilities;
#use DBI;

##############################################################################
# MakeExpressionVectors.pl: This script processes the output of expression#
#                           data                                             #
# from the database. A prerequisite is that the data must be sorted by     #
# At Numbers, where, the first column is the AT number.                    #
# Also puts results into database gene_expression table in pops            #
#                                                                          #
# Input format: <AtNum> <probe ID> <Expt details> <logi value>             #
# Output format: <AtNum_probeID_part_replica> <expt1-value> <expt2-value>.#
# Author: apati@vt.edu                                                     #
# Last Updated: April 15, 2005                                             #
# ##########################################################################

########        CAUTION       #############
#In input file, put control specific data
#first
my $USAGE="$0 <Input Filename> <File of experiment names> <Ouput Filename>";

if(@ARGV < 2) {
  print $USAGE;
  exit;
```

```
}

# Database specific variables
my $db_host='expresso'; my $dbname='pops';
my $user='apati'; my $passwd='**********';
my $conn=DBI->connect("dbi:Pg:dbname=${dbname};
         host=${db_host}", $user, $passwd, {RaiseError => 1});
my $insert_cmd='';
$conn->begin_work();
my $count=0; my $LIMIT=100000;


open(EXPTFILE, $ARGV[1]) or die "Could not open Experiments file $ARGV[1]\n";
open(INFILE, $ARGV[0]) or die "Could not open Input file $ARGV[0]\n";
open(OUTFILE, ">$ARGV[2]") or die "Could not open Output file $ARGV[2]\n";

#Read in all possible experiments into hash
my @exps = <EXPTFILE>;
my %expthash; my %finhash; my %keyList=(); my %listkey=(); my %control;
my $i=1;

foreach my $expt(@exps) {
  $expt=Utilities::trimwhitespace($expt);
  $expthash{$i}=$expt;
  $i++;
}
my $sizeofexps=keys (%expthash);
#Write header to output file
my $j=1; my $k=1;
print OUTFILE "Specifications";
for($k=1; $k<=$sizeofexps; $k++) {
  print OUTFILE "\t$expthash{$k}";
}
print OUTFILE "\n";

my $keyCount=0; my $prevAT='';

while(<INFILE>) {

  my $line=Utilities::trimwhitespace($_);
  my $AtNum=''; my $probeID=''; my $slideID=''; my $expt='';
  my $rep=''; my $inten=''; my $part=''; my $treat='';
  my $timep=0.0; my $ratio=0.0; my $gid=0;
  my @fields=split(/\t/,$line);

  $AtNum=$fields[0]; $probeID=$fields[1]; $inten=$fields[3];
```

```
#Incorporate check for AtNum field being there
if(@fields < 4) {
    $inten=$fields[2]; $fields[2]=$probeID;
    $probeID=$AtNum; $AtNum='X';
}

#print "At=$AtNum, probe=$probeID, inten=$inten, mid=$fields[2]";

#Incorporate 100000 limit check. Write to Output file if necessary
if(($prevAT ne $AtNum)) {
  #Printing to output file
  #Iterate through key list, then expt list
  my ($expkey, $expvalue, $keykey, $keyvalue);
  my $sizeofkeys=keys (%keyList);
  $k=1;
  for($j=1; $j<=$sizeofkeys; $j++) {
    print OUTFILE "$keyList{$j}";
    for($k=1; $k<=$sizeofexps; $k++) {
      #print "Inside Print List\n";
      if(exists $finhash{$keyList{$j},$expthash{$k}}) {
        print OUTFILE "\t$finhash{$keyList{$j},$expthash{$k}}";
      } else {
        print OUTFILE "\tundefined";
      }
    }
    print OUTFILE "\n";
  } # end of for

  #Re-initialize everything
  $keyCount=0;
  %finhash=(); %keyList=(); %listkey=();
} # end of if

#Parse out rest of the parameters
if($fields[2] ne "") {
  my @params=split(/_/,$fields[2]);
  $slideID=$params[1]; $expt=$params[2]; $rep=$params[3];

  if($expt ne "") {
    my @mpars=split(/-/,$expt);
    $expt=$mpars[0]."-".$mpars[2]; $part=$mpars[1];
    $treat=$mpars[0]; $timep=$mpars[2]; $timep=~s/h//g;

    #Populate hash
    my $hashkey=$AtNum."-".$probeID."-".$part."-".$rep;
    #print "Experiment=$expt\n";
```

```
#Take Care of ratio
if($treat=~m/Control/) {
  $control{$probeID,$timep,$part,$rep}=$inten;
  $ratio=1.0;
} else {
  if($control{$probeID,$timep,$part,$rep}==0) {
    $ratio=0.0;
  } else {
    $ratio=$inten/$control{$probeID,$timep,$part,$rep};
  }
  #print "Ratio is $ratio\n";
}
#Check for multiple AT Numbers in the same slide
my @AtArr=split(/;/,$AtNum);
#if(@AtArr>1) { print "Multiple At Numbers: $AtNum\n"; }

for (my $i=0; $i<@AtArr; $i++) {
  #print "ATNUMBER: $AtArr[$i]\n";
  if($AtArr[$i]=~m/ATMG/g) { ; } else {
    $count++;
    if($count>=$LIMIT) {
      $conn->commit();
      $count=0;
      $conn->begin_work();
    }
    #print "Mitochondrial Gene\n";
    #Select gid from gene table
    my $query="SELECT gene_id ";
    $query.="FROM gene ";
    $query.="WHERE gene_locus ilike ";
    $query.="'\%$AtArr[$i]%'";
    my $sth=$conn->prepare($query) or die $conn->errorMessage;
    my $result=$conn->selectall_arrayref($sth);
    while (@$result && (my $row=shift(@$result))) {
      $gid=@$row[0];
      #print "GID for $AtNum is $gid\n";
    }
    #Insert into database also here
    #print "geneid=$gid, slideid=$probeID, tissue=$part,
            treatment=$treat, timepoint=$timep, rep=$rep,
            logi=$inten, ratio=$ratio\n";
    $insert_cmd="insert into gene_expression (gene_id, slide,
                tissue, treatment, timepoint, rep, logi, ratio)
                values ($gid, '$probeID', '$part', '$treat',
                $timep, '$rep', $inten, $ratio)";
```

```
          if($treat=~m/Control/) { ; } else {
            my $s1th=$conn->do($insert_cmd);
          }
        } #End of else
      } #End of for

      #Add hashkey to list of keys
      if(!exists $listkey{$hashkey}) {
        $keyCount=$keyCount+1;
        $listkey{$hashkey}=$keyCount;
        $keyList{$keyCount}=$hashkey;
      }
      $finhash{$hashkey, $expt}=$inten;
    } #end of expt null check
  } #end of fields[2] null check
  $prevAT=$AtNum;
}
#Print last gene record to file
my $sizeofkeys=keys (%keyList);
for($j=1; $j<=$sizeofkeys; $j++) {
  print OUTFILE "$keyList{$j}";
  for($k=1; $k<=$sizeofexps; $k++) {
    if(exists $finhash{$keyList{$j},$expthash{$k}}) {
      print OUTFILE "\t$finhash{$keyList{$j},$expthash{$k}}";
    } else {
      print OUTFILE "\tundefined";
    }
  }
  print OUTFILE "\n";
} # end of for
close(EXPTFILE); close(INFILE); close(OUTFILE);
$conn->commit();
$conn->disconnect if($conn);
```

E.2 Perl Code: CorrelateRepVectors.pl

```
###############################################################################
# CorrelateRepVectors.pl: This script correlates replicate vectors for    #
#                         each <treatment,time-point> tuple               #
# Pre Requisite: Roots and Shoots specific expression vectors must be     #
#                in separte files for correct results                     #
#                                                                         #
# Usage: $0 <ExpressionVectors file> <Experiments File> <Output File>     #
# Author: apati@vt.edu                                                    #
# Last Updated: April 3, 2005                                             #
```

```
# ##############################################################################

#!/usr/bin/perl -w

use strict;
use Vector;
use Utilities;
use Matrix;

my $USAGE="$0 <ExpressionVectors file> <Experiments File> <Output File>";

if(!@ARGV) {
  print "$USAGE\n";
  exit;
}

open(INFILE, $ARGV[0]) or die "CorrelateVectors.pl:
                                Couldn't open input file $ARGV[0]\n";
open(EXPTFILE, $ARGV[1]) or die "CorrelateVectors:
                                  Could not open Experiments file $ARGV[1]\n";
open(OUTFILE, ">$ARGV[2]") or die "CorrelateVectors.pl:
                                    Couldn't open output file $ARGV[2]\n";
my $SNA=0; my $WNA=0; my $NA=0; my $WPA=0; my $SPA=0;
my $rows=0;
my %genehash; my $genecount=0; my %expthash;
my @D2A; my $Dc=0;

#Read in all experiments
my @exps = <EXPTFILE>;
my $i=1;
foreach my $expt(@exps) {
  $expt=Utilities::trimwhitespace($expt);
  $expthash{$i}=$expt;
  $i++;
}
my $sizeofexps=keys (%expthash);

while(<INFILE>) {
  $rows++;
  my $line=Utilities::trimwhitespace($_);
  my @array=split(/\t/,$line);
  my $label=shift(@array);

  #Store specifications-label
  $genecount++;
  $genehash{$genecount}=$label;
```

```
  $D2A[$Dc]=\@array;
  $Dc++;
}

#Strip off last comma and make matrix
my $mat = new Math::Matrix (@D2A);
#$mat->print("MAT\n");

#Calculate Correlations and write to output file
#Experiment hash keys begin at 0.
#Gene hash keys begin at 1
my ($string1,$string2);
print $sizeofexps;
my $avg=0;
for($i=1; $i<=$sizeofexps; $i++) {
  #Extract vectors
  $string1=''; $string2='';
  my $column=$mat->slice($i-1);
  my @array=split(/\n/,$column);
  my $j=0;
  for($j=0; $j<@array; $j++) {
    $array[$j]=Utilities::trimwhitespace($array[$j]);
    if($j % 2 == 0) {
      $string2=$string2.$array[$j]."_";
    } elsif($j % 2 ==1) {
        $string1=$string1.$array[$j]."_";
    }
  }
  #Calculate Spearman Correlation
  chop($string1); chop($string2);
  #my $rho_coeff=Vector::Spearman_Correlation($string1,"_",$string2,"_");
  my $rho_coeff=Vector::Pearson_Correlation($string1,"_",$string2,"_");
  $avg=$avg+$rho_coeff;
  print "\n***print $rho_coeff";
  if(-1<=$rho_coeff && $rho_coeff<-0.7) {
    $SNA++;
  } elsif(-0.7<=$rho_coeff && $rho_coeff<-0.3){
    $WNA++;
  } elsif(-0.3<=$rho_coeff && $rho_coeff<0.3){
    $NA++;
  } elsif(0.3<=$rho_coeff && $rho_coeff<0.7){
    $WPA++;
  } elsif(0.7<=$rho_coeff && $rho_coeff<=1.0){
    $SPA++;
  }
  print OUTFILE "$expthash{$i}\t$rho_coeff\n";
```

```
}
$avg=$avg/$sizeofexps;
print "\n*****************************";
print "\nPrinting Correlation Summary";
print "\n*****************************\n";
print "\nAverage Correlation Coefficient: $avg";
print "\nNumber of Strong Negative Correlations: $SNA\n
      Number of Weak Negative Correlations: $WNA\n
      Number of Little or No Correlations: $NA\n
      Number of Weak Positive Correlations: $WPA\n
      Number of Strong Positive Correlations: $SPA\n";
close(INFILE); close(OUTFILE); close(EXPTFILE);
```

E.3 Perl Code: MakeAverageVectors.pl

```
#############################################################################
# MakeAverageVectors.pl: This script find average vectors of both         #
#                        replicates of each gene with treated/control     #
#                        ratios as vector values                          #
#                         each <treatment,time-point> tuple               #
#                                                                          #
# Usage: $0 <File with Rep Vectors> <Output file for average vectors>     #
# Author: apati@vt.edu                                                     #
# Last Updated: May 31, 2005                                               #
# ##########################################################################

#!/usr/bin/perl

use strict;
use Utilities;
use Vector;
use Shell;

my $USAGE="$0 <File with Rep Vectors> <Output file for average vectors>\n";

if(!@ARGV) {
  print "$USAGE";
  exit;
}
open(INFILE1,$ARGV[0]) or die "Couldn't open input file $ARGV[0]\n";
open(OUTFILE,">$ARGV[1]") or die "Couldn't open output file $ARGV[1]\n";

#Make initial vectors hash
my %vechash; my %Athash;my $i=0;
while(<INFILE1>) {
  $i++; #print "Line Number: $i\n";
```

```
  my $line = Utilities::trimwhitespace($_);
  my @array=split(/\t/,$line);
  my $label=shift(@array);
  #Parse out parameters
  my @pars=split(/-/,$label);
  my $AtNum=$pars[0]; my $slide_ID=$pars[1];
  my $part=$pars[2]; my $rep=$pars[3];
  #print "$AtNum\n";
  if($AtNum eq 'X') {
  #print "No AT number\n";
    $vechash{$slide_ID,$rep}=join('_',@array);
    if(!exists $Athash{$slide_ID}) {
      $Athash{$slide_ID}=1;
    }
  } else {
    my @ATarr=split(/;/,$AtNum);
    foreach my $num(@ATarr) {
      chomp($num);
      $vechash{$num,$rep}=join('_',@array);
      if(!exists $Athash{$num}) {
        $Athash{$num}=1;
      }
    }
  }
}
my $vechashsize= keys(%vechash); my $Atsize= keys(%Athash);
print "Number of vectors=$vechashsize, Number of ATs=$Atsize\n";
#Make average vectors hash
my %avgVecs;
while ((my $key, my $val)=each(%Athash)) {
  my @vec1=split(/_/,$vechash{$key, "Rep1"});
  my @vec2=split(/_/,$vechash{$key, "Rep2"});
  my $av=Vector::Vector_Average(\@vec1,\@vec2);
  #print "@$av\n";
  $avgVecs{$key}=join('_',@$av);
  my $finvec=Scale($avgVecs{$key},"_");
  my @arr=split(/_/,$finvec);
  print OUTFILE "$key\t@arr\n";
}
my $avgvecsize=keys(%avgVecs);
print "Average Vector Size: $avgvecsize\n";
#delete vector hash
%vechash=();
close(INFILE1); close(OUTFILE);
```

```
sub Scale($$) {
  my $line=$_[0]; my $sep=$_[1];
  my @array=split(/$sep/,$line);
  my @arrcopy=split(/$sep/,$line);
  my $i=0;
  my @divarray;
  for(my $j=0; $j<5; $j++) {
    $divarray[$j]=shift(@array);
    $divarray[$j]=shift(@arrcopy);
  }
  while($array[$i]) {
    for(my $j=0; $j<5; $j++) {
      #$array[$i]=$array[$i]/$divarray[$j];
      $array[$i]=log($array[$i])-log($divarray[$j]);
      $i++;
    }
  }
  my $res=join('_',@array);
  #print "\n@array\n";
  return $res;
}
```

E.4 Perl Code: GeneCorrelate.pl

```
##############################################################################
# GeneCorrelate.pl: This script finds correlations within a geneset       #
#                   or between a geneset and the rest of the genome       #
# Usage: As described in code                                             #
# Author: apati@vt.edu                                                    #
# Last Updated: May 31, 2005                                              #
# ##########################################################################

#!/usr/bin/perl

use strict;
use Utilities;
use Vector;
use Shell;

my $USAGE = "$0\nArg1-<file with all expression vectors>\n
                 Arg2-<file with gene cluster At nums>\n
                 Arg3-<File with experiments in order of columns>\n
                 Arg4-<output file of correlations>\n
                 Arg5-<Use treated/Control ratio? 1=Yes, 0=No>\n
                 Arg6-<correlation-choice? 0=Among genes in list,
```

```
                                        1=With entire genome>\n
                    Arg7-<Confidence level for sig of correlation>\n
                    Arg8-<File to put exp vectors for input to TS>\n
                    Arg9-<threshold-for-selection-from-genome-if Arg6=1>\n
                    Arg10-<If Arg6=1, calculate correlation with difference
                           set from genome? 1=Yes, 0=No>\n";

if(!@ARGV) {
  print "$USAGE";
  exit;
}
open(INFILE1,$ARGV[0]) or die "Couldn't open input file $ARGV[0]\n";
open(INFILE2,$ARGV[1]) or die "Couldn't open input file $ARGV[1]\n";
open(INFILE3,$ARGV[2]) or die "Couldn't open input file $ARGV[2]\n";
open(OUTFILE,">$ARGV[3]") or die "Couldn't open output file $ARGV[3]\n";
my $ratio=$ARGV[4];
my $choice=$ARGV[5];
my $corr_sig=$ARGV[6];
my $GeneVectors=$ARGV[7];
print "Choice=$choice\n";

open(VECTORFILE, ">$GeneVectors");
print VECTORFILE "# title\n";
print VECTORFILE "Expression vectors of cluster of gene\n";
print VECTORFILE "# static attributes\n";
print VECTORFILE "Name,String\n";
print VECTORFILE "# Dynamic Attributes\n";
print VECTORFILE "logi,Float\n";

#Make Experiment Hash
my @exps = <INFILE3>;
my $i=1; my %expthash;
foreach my $expt(@exps) {
  $expt=Utilities::trimwhitespace($expt);
  $expthash{$i}=$expt;
  $i++;
}
my $sizeofexps=keys (%expthash);

print VECTORFILE "# of time points=n\n";
my $numtp=$sizeofexps-5;
print VECTORFILE "$numtp\n";

my @AtNums = <INFILE2>;

if($choice==0) {
```

```
#Make initial vectors hash
my %vechash;
while(<INFILE1>) {
  my $line = Utilities::trimwhitespace($_);
  foreach my $At(@AtNums) {
    $At=Utilities::trimwhitespace($At);
    #The line matches one of the At Numbers. Populate Hashes
    #print $At;
    if($line=~m/$At/gi) {
      my @array=split(/\t/,$line);
      my $label=shift(@array);
      #Parse out parameters
      my @pars=split(/-/,$label);
      my $AtNum=$pars[0]; my $slide_ID=$pars[1];
      my $part=$pars[2]; my $rep=$pars[3];
      #$vechash{$AtNum,$rep}=join('_',@array);
      #handle the case of multiple AT numbers
      my @ATarr=split(/;/,$AtNum);
      foreach my $num(@ATarr) {
        chomp($num);
        $vechash{$num,$rep}=join('_',@array);
      }
    }
  }
}

#Eliminate AT numbers for which there was no expression data
my @AtNumnew; $i=0;
foreach my $At(@AtNums) {
  if(!exists $vechash{$At,"Rep1"}) {
    print "\nLog Message: $At doesn't have expression data
           \nDeleting $At from list\n";
  }
  else {
    $AtNumnew[$i]=$At; $i++;
  }
}

my $numRecords=@AtNumnew;
print VECTORFILE "# of records k\n";
print VECTORFILE "$numRecords\n";

print VECTORFILE "# time point labels\n";
my @tps = splice @exps, 5, 45;
my $tpspr=join(",",@tps);
print "***$tpspr\n";
```

```
  print VECTORFILE "$tpspr\n";

  #Make Avg vector Hash
  my %avgVecs;
  foreach my $At(@AtNumnew) {
    my @vec1 = split(/_/,$vechash{$At,"Rep1"});
    my @vec2 = split(/_/,$vechash{$At,"Rep2"});
    #print "Vector1:@vec1\n"; print "Vector2:@vec2\n";
    my $avg = Vector::Vector_Average(\@vec1,\@vec2);
    my @avg=@$avg; #print "@avg";
    $avgVecs{$At} = join('_',@avg);
    #print "Average:$avgVecs{$At}\n";
  }

  %vechash=();
  #my %corrhash;
  #Find Pairwise Correlations
  foreach my $At1(@AtNumnew) {
    my $vec1;
    if($ratio==1) {
      $vec1=Scale($avgVecs{$At1},"_");
    } else {
      $vec1=$avgVecs{$At1};
    }

    #Print to vectorfile
    my $printVec=$vec1;
    $printVec=~s/_/,/g;
    print VECTORFILE "$At1,$printVec\n";

    foreach my $At2(@AtNumnew) {
      my $vec2;
      if($ratio==1) {
        $vec2=Scale($avgVecs{$At2},"_");
      } else {
        $vec2=$avgVecs{$At2};
      }
      print "$At1\t$At2\n";
      my $corr=Vector::All_Correlations($vec1,"_",$vec2,"_");
      #my $corr=Vector::Spearman_Correlation($vec1,"_",$vec2,"_");
      print OUTFILE "$At1\t$At2\t$corr\n";
    }
  }
}
close(VECTORFILE);
if($choice==1) { #Find correlation of each gene with rest of the genome genes
```

```
my $diff=$ARGV[9];
my $threshold=$ARGV[8];
print "Threshold=$threshold\n";

#Extract vector corresponding to genes first
my %genlist;
#my @genarray=<INFILE2>;
print "Printing array: @AtNums\n";
foreach my $gene(@AtNums) {
  $gene=Utilities::trimwhitespace($gene);
  qx(grep $gene $ARGV[0] > "$gene.txt");
  open(GENFILE,"$gene.txt") or die "CorrelateGene.pl:
                                    Couldn't find genewise file $gene.txt\n";
  my @garr=<GENFILE>;
  my $rep2v=$garr[0]; my $rep1v=$garr[1];
  my @vec1=split(/\t/,$rep1v); shift(@vec1);
  my @vec2=split(/\t/,$rep2v); shift(@vec2);
  my $vecavg=Vector::Vector_Average(\@vec1,\@vec2);
  print "Avg Vector: @$vecavg\n";
  $genlist{$gene}=join('_',@$vecavg);
  close(GENFILE);
  rm("$gene.txt");
}

#Make initial vectors hash
my %vechash; my %Athash;my $i=0;
while(<INFILE1>) {
  $i++; #print "Line Number: $i\n";
  my $line = Utilities::trimwhitespace($_);
  my @array=split(/\t/,$line);
  my $label=shift(@array);
  #Parse out parameters
  my @pars=split(/-/,$label);
  my $AtNum=$pars[0]; my $slide_ID=$pars[1];
  my $part=$pars[2]; my $rep=$pars[3];
  #print "$AtNum\n";
  if($AtNum eq 'X') {
    #print "No AT number\n";
    $vechash{$slide_ID,$rep}=join('_',@array);
    if(!exists $Athash{$slide_ID}) {
      $Athash{$slide_ID}=1;
    }
  } else {
    my @ATarr=split(/;/,$AtNum);
    foreach my $num(@ATarr) {
      chomp($num);
```

```
      $vechash{$num,$rep}=join('_',@array);
      if(!exists $Athash{$num}) {
        $Athash{$num}=1;
      }
    }
  }
}
my $vechashsize= keys(%vechash); my $Atsize= keys(%Athash);
print "Number of vectors=$vechashsize, Number of ATs=$Atsize\n";
#Make average vectors hash
my %avgVecs;
while ((my $key, my $val)=each(%Athash)) {
  my @vec1=split(/_/,$vechash{$key, "Rep1"});
  my @vec2=split(/_/,$vechash{$key, "Rep2"});
  my $av=Vector::Vector_Average(\@vec1,\@vec2);
  #print "@$av\n";
  $avgVecs{$key}=join('_',@$av);
}
my $avgvecsize=keys(%avgVecs);
print "Average Vector Size: $avgvecsize\n";
#delete vector hash
%vechash=();

#Iterate through average hash
while ((my $key, my $val)=each(%Athash)) {
  my $vec1=$avgVecs{$key};
  while ((my $gkey, my $gval)=each(%genlist)) {
    #check if difference gene-list is desired
    my $flag;
    if(exists($genlist{$key})) { $flag=0; } else { $flag=1; }

    if($flag==0 && $diff==1) { ; } else {
      my ($v1, $v2);
      if($ratio==1) {
        $v1=Scale($vec1,"_"); $v2=Scale($gval,"_");
      } else {
        $v1=$vec1; $v2=$gval;
      }
      #my $corr=Vector::Spearman_Correlation($v1,"_",$v2,"_",$corr_sig);
      #my $pcorr=Vector::Pearson_Correlation($v1,"_",$v2,"_");
      my $corr=Vector::All_Correlations($v1,"_",$v2,"_");
      #print "Corr=$corr\n";
      if($corr>=$threshold) {
        #print "$v1\n$v2\n";
        print OUTFILE "$gkey\t$key\t$corr\n";
      }
```

```
      }#end of if

    }#end of while
  }
}

close INFILE1; close INFILE2; close INFILE3; close OUTFILE;


#################################################################
# Scale: Subroutine for evaluating treated to control ratios
#################################################################
sub Scale($$) {
  my $line=$_[0]; my $sep=$_[1];
  my @array=split(/$sep/,$line);
  my @arrcopy=split(/$sep/,$line);
  my $i=0;
  my @divarray;
  for(my $j=0; $j<5; $j++) {
    $divarray[$j]=shift(@array);
    #$divarray[$j]=shift(@arrcopy);
  }
  while($array[$i]) {
    for(my $j=0; $j<5; $j++) {
      #$array[$i]=$array[$i]/$divarray[$j];
      $array[$i]=log($array[$i])-log($divarray[$j]);
      $i++;
    }
  }
  my $res=join('_',@array);
  #print "\n@array\n";
  return $res;
}
```

E.5 Perl Code: SelectTreatmentVectors.pl

```
#################################################################################
# SelectTreatmentVectors.pl: This script creates exppression vectors      #
#                            that are subsets of the entire 45 long       #
#                            expression vector                            #
# Usage: As described in code                                             #
# Author: apati@vt.edu                                                    #
# Last Updated: June 8, 2005                                              #
# ##############################################################################
#!/usr/bin/perl
```

```
use strict;

my $USAGE="$0\nARG1: File with all Expression Vectors of length 45, ratioed\n
               ARG2: File with Treatment Coordinates\n
               ARG3: File with treatments to be selected\n
               ARG4: Output File of Vectors\n
               CAUTION: Order is Important!! Treatments in the selection file
               MUST be in the same order as in the Treatment Coordinates file\n";

if(!@ARGV) {
  print "$USAGE";
  exit;
}

open(INFILE1,$ARGV[0]) or die "Couldn't open input file $ARGV[0]\n";
open(INFILE2,$ARGV[1]) or die "Couldn't open input file $ARGV[1]\n";
open(INFILE3,$ARGV[2]) or die "Couldn't open input file $ARGV[2]\n";
open(OUTFILE,">$ARGV[3]") or die "Couldn't open output file $ARGV[3]\n";

my %treatstart; my %treatend;

while(<INFILE2>) {
  chomp($_);
  my @arr=split(/\t/,$_);
  $treatstart{$arr[0]}=$arr[1];
  $treatend{$arr[0]}=$arr[2];
}

my @selection=<INFILE3>;

while(<INFILE1>) {
  chomp($_);
  my @arr=split(/\t/,$_);
  my $newvec='';

  foreach my $choice(@selection) {
    chomp($choice);
    my @vector=split(/ /,$arr[1]);
    my $index=$treatstart{$choice}-1;
    #print "Index=$index\n";
    my $temp=join(' ',splice(@vector,$index,5));
    if($newvec eq '') {
      $newvec=$temp;
    } else {
      $newvec=$newvec.' '.$temp;
    }
```

```
  }

  print OUTFILE "$arr[0]\t$newvec\n";
}



close(INFILE1); close(INFILE2); close(INFILE3); close(OUTFILE);
```

E.6 MATLAB Code: GeneStat.m

```
//Find Correlation statistics
function GeneStat(AllVecs,AtNums,SelVecs,Selection,CorrData,
                  PCorrData,KCorrData,outStat,outPVal)

fos=fopen(outStat,'w');
fop=fopen(outPVal,'w');

MeanArr=[]; % Array with Mean Values of rho's distribution
StdArr=[]; % Array with Standard Deviation Values of rho's distribution
PMArr=[]; % Array with Mean Values of pearson r
PStdArr=[]; % Array with STDEV Values of perason r
KMArr=[]; % Array with Mean Values of Kendall tau
KStdArr=[]; % Array with STDEV Values of Kendall tau

Coeffs=[]; PCoeffs=[]; KCoeffs=[];

% Get Correlation Values with the rest of the genome for each gene
[NUMSEL,SIZESEL]=size(SelVecs);
[NUMTOT,SIZETOT]=size(AllVecs);

for i=1:NUMSEL
    X=(SelVecs(i,:))';
    [At1,err]=sprintf('%s',Selection(i,:));
    Coeffs=[];
   for j=1:NUMTOT
       Y=(AllVecs(j,:))';
       [srho,pval]=corr(X,Y,'type','spearman');
       Coeffs(j)=srho;
       [pr,pval]=corr(X,Y,'type','pearson');
       PCoeffs(j)=pr;
       [ktau,pval]=corr(X,Y,'type','kendall');
       KCoeffs(j)=ktau;
   end
   MeanArr(i)=mean(Coeffs);PMArr(i)=mean(PCoeffs);KMArr(i)=mean(KCoeffs);
```

```
   StdArr(i)=std(Coeffs);PStdArr(i)=std(PCoeffs);KStdArr(i)=std(KCoeffs);
   fprintf(fos,'%s\t%f\t%f\t%f\t%f\t%f\t%f\n',At1,MeanArr(i),StdArr(i),
           PMArr(i),PStdArr(i),KMArr(i),KStdArr(i));
end

% Calculate p-values for each correlation
for i=1:NUMSEL
   for j=1:NUMSEL
        if(i ~= j)
            [At1,err]=sprintf('%s',Selection(i,:));
            [At2,err]=sprintf('%s',Selection(j,:));
            zpval=1-normcdf(CorrData(i,j),MeanArr(i),StdArr(i));
            Pzpval=1-normcdf(PCorrData(i,j),PMArr(i),PStdArr(i));
            Kzpval=1-normcdf(KCorrData(i,j),KMArr(i),KStdArr(i));
            fprintf(fop,'%s\t%s\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n',At1,At2,PCorrData(i,j),
                    Pzpval,CorrData(i,j),zpval,KCorrData(i,j),Kzpval);
        end
   end
end


fclose(fos);
fclose(fop);
return;
```

E.7 MATLAB Code: GeneCorrelateInter.m

```
//Correlate a genelist within itself
function GeneCorrelateInter (infileAll,infileList,outfile,part,outstat,outpvals)

% Read in All Expression Vectors
fin=fopen(infileAll,'r');
flist=fopen(infileList,'r');
fout=fopen(outfile,'w');
A=[];
AtNums=[]; % Contains all the AT numbers
AllVecs=[]; % Contains all the expression vectors
Selection=[]; % Cotains AT numbers to be compared against the rest of the genome
SelVecs=[]; % Contains all the selection expression vectors
CorrData=[]; % Contains pairwise correlations between genes
PCorrData=[];
KCorrData=[];
veccnt=0;
tic

selcnt=0;
```

```
while(flist)
   line=fgetl(flist);
   selcnt=selcnt+1;
   Selection(selcnt,:)=line;
   eofstat = feof(flist);
   if(eofstat==1)
    break;
   end
end

[SELGENES, m]=size(Selection)


while(fin)
    line=fgetl(fin);
    [AtNum, str]=strtok(line);
    A=str2num(str);
    veccnt=veccnt+1;
    AtNums(veccnt,:)=AtNum;
    AllVecs(veccnt,:)=A;

    % Check if it is part of the selection
    for i=1:SELGENES
        [s,err]=sprintf('%s',Selection(i,:));
        res=strcmp(AtNum,s);
        if(res == 1)
            SelVecs(i,:)=A;
        end
    end

    eofstat = feof(fin);
    if(eofstat==1)
        break;
    end
end
[TOTALGENES, n]=size(AllVecs)

%[s, err]=sprintf('%s', AtNums(1,:));

for i=1:SELGENES
    X=(SelVecs(i,:))';
    [At1,err]=sprintf('%s',Selection(i,:));
    for j=(i+1):SELGENES
        Y=(SelVecs(j,:))';
        [At2,err]=sprintf('%s',Selection(j,:));
        if(strcmp(At1,At2)==0)
```

```
            [rho,p]=corr(X,Y,'type','Spearman');
            fprintf(fout,'%s\t%s\t%f\n',At1,At2,rho);
            CorrData(i,j)=rho; CorrData(j,i)=rho;
            [r,p]=corr(X,Y,'type','pearson');
            PCorrData(i,j)=r; PCorrData(j,i)=r;
            [tau,p]=corr(X,Y,'type','kendall');
            KCorrData(i,j)=tau; KCorrData(j,i)=tau;
        end
    end
 end

t_toc=toc
[a,b]=size(CorrData)
GeneStat(AllVecs,AtNums,SelVecs,Selection,CorrData,PCorrData,
         KCorrData,outstat,outpvals);
fclose('all');
```

E.8 MATLAB Code: GeneCorrelateWithGenome.m

```
//Correlate a genelist with the rest of the genome
function GeneCorrelateWithGenome (infileAll,infileList,cutOff,outfile,part)

% Read in All Expression Vectors
fin=fopen(infileAll,'r');
flist=fopen(infileList,'r');
fout=fopen(outfile,'w');
A=[];
AtNums=[]; % Contains all the AT numbers
AllVecs=[]; % Contains all the expression vectors
Selection=[]; % Cotains AT numbers to be compared against the rest of the genome
SelVecs=[]; % Contains all the selection expression vectors
veccnt=0;
tic

selcnt=0;
while(flist)
   line=fgetl(flist);
   selcnt=selcnt+1;
   Selection(selcnt,:)=line;
   eofstat = feof(flist);
   if(eofstat==1)
    break;
   end
end

[SELGENES, m]=size(Selection)
```

```
while(fin)
    line=fgetl(fin);
    [AtNum, str]=strtok(line);
    A=str2num(str);
    veccnt=veccnt+1;
    AtNums(veccnt,:)=AtNum;
    AllVecs(veccnt,:)=A;

    % Check if it is part of the selection
    for i=1:SELGENES
        AtNum
        [s,err]=sprintf('%s',Selection(i,:));
        res=strcmp(AtNum,s);
        if(res == 1)
            SelVecs(i,:)=A;
        end
    end

    eofstat = feof(fin);
    if(eofstat==1)
        break;
    end
end
[TOTALGENES, n]=size(AllVecs)

%[s, err]=sprintf('%s', AtNums(1,:));

for i=1:SELGENES
    X=(SelVecs(i,:))';
    [At1,err]=sprintf('%s',Selection(i,:));
    for j=1:TOTALGENES
        Y=(AllVecs(j,:))';
        [At2,err]=sprintf('%s',AtNums(j,:));
        if(strcmp(At1,At2)==0)
            [rho,p]=corr(X,Y,'type','Spearman');
            if(rho > cutOff)
                fprintf(fout,'%s\t%s\t%f\n',At1,At2,rho);
            end
        end
    end
 end

t_toc=toc
```

```
fclose('all');
```

E.9 Perl Code: copyCorrstoDB.pl

```
#!/usr/bin/perl

##############################################################################
# copyCorrstoDB.pl: This script copies gene pair correlations into        #
#                   the expr_correlations table in pops database          #
# Usage: As described in code                                             #
# Author: apati@vt.edu                                                    #
# Last Updated: June 8, 2005                                              #
# ##########################################################################

use strict;
use Utilities;
use Vector;
use Shell;
use DBI;

# Database specific variables
my $db_host='expresso'; my $dbname='pops';
my $user='apati'; my $passwd='**********';
my $conn=DBI->connect("dbi:Pg:dbname=${dbname};host=${db_host}",
                      $user, $passwd, {RaiseError => 1});
my $insert_cmd='';

my $USAGE="$0 <input-file> <file-with-experiments> <part>\n";
if(!@ARGV) {
  print "$USAGE";
  exit;
}
open(INFILE1,$ARGV[0]) or die "GeneCorrelate.pl:
                               Couldn't open input file $ARGV[0]\n";
open(INFILE3,$ARGV[1]) or die "GeneCorrelate.pl:
                               Couldn't open input file $ARGV[2]\n";
my $part=$ARGV[2];
my @exps = <INFILE3>;
my $i=1; my %expthash; my @arrayset; my $j=0;
foreach my $expt(@exps) {
  $expt=Utilities::trimwhitespace($expt);
  $expthash{$i}=$expt;
  #Datbase Specific
    if(!($expt=~m/Control/)) {
      my @ear=split(/-/,$expt); my $tmpt=$ear[1]; my $tiss=$ear[2];
```

```
      my @ear1=split(/stress/,$ear[0]); my $stress=$ear1[0];
      my $selq="SELECT uid from arrays where treatment ilike '%$stress%'
                and timepoint='$tmpt' and tissue='$tiss'";
      my $sth=$conn->prepare($selq) or die $conn->errorMessage;
      my $result=$conn->selectall_arrayref($sth);
      while (@$result && (my $row=shift(@$result))) {
        #print "Query=$selq;Array=@$row[0]\n";
        $arrayset[$j]=@$row[0]; $j++;
      }
    }
    $i++;
}
my $sizeofexps=keys (%expthash);
my $arrays=join('_',@arrayset);
#Check if the record exists in the database
my $aq3="select uid from array_set where members='$arrays'";
my $ath=$conn->prepare($aq3) or die $conn->errorMessage;
my $ares=$conn->selectall_arrayref($ath);
if (!(@$ares && (my $row=shift(@$ares)))) {
  #Insert into arrayset table and get id
  print "Inserting New ArraySet into table\n";
  $insert_cmd="INSERT into array_set (members) values ('$arrays')";
  $conn->do($insert_cmd);
} else {
  print "ArraySet already exists. No new entries added to array_set table\n";
}
my $selq="SELECT uid from array_set where members='$arrays'";
my $s7th=$conn->prepare($selq) or die $conn->errorMessage;
my $res7=$conn->selectall_arrayref($s7th);
my $arrsetid;
while (@$res7 && (my $row=shift(@$res7))) {
  $arrsetid=@$row[0];
}
print "Array Set is $arrsetid\n";

while(<INFILE1>) {
  my $line=$_; chomp($line); my @params=split(/\t/,$line);
  my $Atn1=Utilities::trimwhitespace($params[0]);
  my $Atn2=Utilities::trimwhitespace($params[1]);
  my @arr1=split(/;/,$Atn1); my @arr2=split(/;/,$Atn2);

  foreach my $At1(@arr1) {
    foreach my $At2(@arr2) {

      #Find gid
      my $gid1;
```

```
      my $sq1="SELECT gene_id from gene WHERE gene_locus ilike '\%$At1%'";
      my $sth=$conn->prepare($sq1) or die $conn->errorMessage;
      my $result=$conn->selectall_arrayref($sth);
      while (@$result && (my $row=shift(@$result))) {
        $gid1=@$row[0];
        if ($gid1=='') { print "Couldn't retrieve gid for $At1\n"; }
        #print "GID for $At1 is $gid1\n";
      }

      #Find gid
      my $gid2;
      my $sq2="SELECT gene_id from gene WHERE gene_locus ilike '\%$At2%'";
      my $s2th=$conn->prepare($sq2) or die $conn->errorMessage;
      $result=$conn->selectall_arrayref($s2th);
      while (@$result && (my $row=shift(@$result))) {
        $gid2=@$row[0];
        if ($gid2=='') { print "Couldn't retrieve gid for $At2\n"; }
        #print "GID for $At2 is $gid2\n";
      }

      if($gid1!='' && $gid2!='') {
        #Check if the record exists in the database
        my $sq3="select uid from expr_correlations where
                 ((gene1='$gid1' and gene2='$gid2') or
                  (gene2='$gid1' and gene1='$gid2')) and
                 tissue='$part' and array_set='$arrsetid'";
        my $s3th=$conn->prepare($sq3) or die $conn->errorMessage;
        $result=$conn->selectall_arrayref($s3th);
        if (!(@$result && (my $row=shift(@$result)))) {
          #Insert into database
          #print "Inserting into database\n";
          $insert_cmd="INSERT into expr_correlations (
                       gene1,gene2,pearson_r,pearson_significance,spearman_rho,
                       spearman_significance,kendall_tau,kendall_significance,
                       tissue,array_set) values
                       ($gid1,$gid2,$params[2],$params[3],$params[4],$params[5],
                        $params[6],$params[7],'$part',$arrsetid)";
          $conn->do($insert_cmd);
        } else {
          #print "Record Exists. No insert done\n";
        }
      }#end of null-check for gids
    }#end of inner foreach
  }#end of outer foreach
}#end of while INFILE
close(INFILE); close(INFILE3);$conn->disconnect if $conn;
```

E.10 MATLAB Code: CallSimulate.m

```
function CallSimulate (infile,outfile,part)

% Read in All Expression Vectors
fin=fopen(infile,'r');
fout=fopen(outfile,'w');
A=[];
AtNums=[]; % Contains all the AT numbers
AllVecs=[]; % Contains all the expression vectors
veccnt=0;
tic
while(fin)
    line=fgetl(fin);
    [AtNum, str]=strtok(line);
    A=str2num(str);
    veccnt=veccnt+1;
    AtNums(veccnt,:)=AtNum;
    AllVecs(veccnt,:)=A;
    eofstat = feof(fin);
    if(eofstat==1)
        break;
    end
end

[TOTALGENES, n]=size(AllVecs)

%[s, err]=sprintf('%s', AtNums(1,:));

index=1;

% Do it for entire matrix
expo=1; index=1; indexend=45; samplesize=13;
while(1)
    %samplesize=power(2,expo)
    ss=num2str(samplesize)
    ofile=strcat('S-Values-Shoots-1-7-',ss,'.txt');
    [Mean_of_Stat,Std_of_Stat]=simulate(AllVecs,100000,samplesize,ofile);
    fprintf(fout,'%s\t%d\t%d\t%d\t%f\t%f\n',part,samplesize,index,
            indexend,Mean_of_Stat,Std_of_Stat);
    samplesize=samplesize+1;
    if(samplesize>15) % Change this
        break;
    end
end

t_toc=toc
```

```
fclose('all');
```

E.11 MATLAB Code: simulate.m

```
function [Mean_of_Stat,Std_of_Stat] = simulate(AllVecs,NUMSAMPLES,
                                                SAMPLESIZE,OBSFILE)

% NUMSAMPLES is the number of times sampling in done
% SAMPLESIZE is the number of randomly selected pairs
%            of genes that are correlated
% infile is the input file with all average expression vectors

[TOTALGENES, n]=size(AllVecs)
fp=fopen(OBSFILE,'w');
%[s, err]=sprintf('%s', AtNums(1,:));

% Start timing Simulation
t=clock;
tic

% Simulate Random Samples
simcount=0; StatVec=[];

for i=1:NUMSAMPLES

    SampCorrVec=[]; sampcorrcnt=0;

    randMat=[];
    for j=1:SAMPLESIZE
        while(1)
            % Randomly select matrix row indices from a uniform distribution
            index=round(1+(TOTALGENES-1)*rand);
            if((index>=1)&&(index<=TOTALGENES))
                break;
            end
        end
        randMat(j,:)=AllVecs(index,:);
    end
    mysum=0;
    for m=1:SAMPLESIZE
        for n=(m+1):SAMPLESIZE
            sampcorrcnt=sampcorrcnt+1;
            v1=(randMat(m,:))'; v2=(randMat(n,:))';
            [SampCorrVec(sampcorrcnt), p]=corr(v1,v2,'type','Spearman');
            val=abs(SampCorrVec(sampcorrcnt)); SampCorrVec(sampcorrcnt)=val;
```

```
            mysum=mysum+val;
        end
    end
    simcount=simcount+1;
    StatVec(simcount)=mysum;
    fprintf(fp,'%f\n',mysum);
end
%StatVec
Mean_of_Stat=mean(StatVec);
Std_of_Stat=std(StatVec);

t_toc=toc
difftime=etime(clock,t)
fclose(fp);
return;
```

E.12 Perl Code: getSDistribution.pl

```
#!/usr/bin/perl

use strict;

my $USAGE="$0 <input file-rootname> <output file> <least samplesize>
           <highest samplesize> <treatment_set> <Part>\n";

if(@ARGV<4) {
  print "$USAGE";
  exit;
}

my $outfilename=$ARGV[1];
for (my $z=$ARGV[2];$z<=$ARGV[3];$z++) {
  my $infilename=$ARGV[0]."$z".".txt";
  buildDist($infilename,$outfilename,$z,$ARGV[4],$ARGV[5]);
}

sub buildDist($$$$$) {
  my $USAGE="$0 <input file> <output file> <numgenes>\n";

  #open(INFILE,$ARGV[0]) or die "Couldn't open input file $ARGV[0]\n";
  #open(OUTFILE,">$ARGV[1]") or die "Couldn't open output file $ARGV[1]\n";
  open(INFILE,$_[0]) or die "Couldn't open input file $_[0]\n";
  open(OUTFILE,">>$_[1]") or die "Couldn't open output file $_[1]\n";

  my $ll=0; my $ul=0.05;
  my %disthash;
```

```
  #my $limit=fac($ARGV[2])/(2*fac($ARGV[2]-2));
  my $limit=fac($_[2])/(2*fac($_[2]-2));

  while(<INFILE>) {
    my $line=$_; chomp($_);
    my $i=$ll; my $j=$ul;
    while(1) {
      if($j>$limit) { last; }
      if($i<=$line && $j>$line) {
        if(exists $disthash{$i,$j}) {
          $disthash{$i,$j}=$disthash{$i,$j}+1;
        } else {
          $disthash{$i,$j}=1;
        }
      } # end of if
      $i=$i+0.05; $j=$j+0.05;
    } # end of while
  }

  my $i=$ll; my $j=$ul;
  my $cdf=0;
  while($j<=$limit) {
    my $count=0;
    if(exists $disthash{$i,$j}) {
      $count=$disthash{$i,$j};
    }
    my $pdf=$count/100000;
    $cdf=$cdf+$pdf;
    printf(OUTFILE "%s\t%d\t%d\t%5.2f\t%5.2f\t%d\t%1.9f\t%1.9f\n",
                    $_[4],$_[3],$_[2],$i,$j,$count,$pdf,$cdf);
    $i=$i+0.05; $j=$j+0.05;  }
}


close(INFILE); close(OUTFILE);
sub fac { $_[0]>1?$_[0]*fac($_[0]-1):1; }
```

Vita

Amrita Pati was born in Bhubaneshwar, India, in 1982. She obtained her B.E (Hons) in Computer Science from Birla Institite of Technology and Science, Pilani, India, in 2003. In August 2003, she began graduate studies at Virginia Tech. The work reported in this thesis was done between September 2004 and July 2005. Her research interests include algorithms, graphs and networks, and bioinformatics.