# GENE EXPRESSION MICROARRAY ANALYSIS USING PATTERN DISCOVERY

A Thesis Submitted to the College of
Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon, Canada

By
Matthew Neil Bainbridge

# PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5A9

# ABSTRACT

Analysis of gene expression microarray data has traditionally been conducted using hierarchical clustering. However, such analysis has many known disadvantages and pattern discovery (PD) has been proposed as an alternative technique. In this work, three similar but different PD algorithms – Teiresias, Splash and Genes@Work – were benchmarked for time and memory efficiency on a small yeast cell-cycle data set. Teiresias was found to be the fastest, and best over-all program. However, Splash was more memory efficient. This work also investigated the performance of four methods of discretizing microarray data: sign-of-the-derivative, K-means, pre-set value, and Genes@Work stratification. The first three methods were evaluated on their predisposition to group together biologically related genes. On a yeast cell-cycle data set, sign-of-the-derivative method yielded the most biologically significant patterns, followed by the pre-set value and K-means methods. K-means, preset-value, and Genes@Work were also compared on their ability to classify tissue samples from diffuse large b-cell lymphoma (DLBCL) into two subtypes determined by standard techniques. The Genes@Work stratification method produced the best patterns for discriminating between the two subtypes of lymphoma. However, the results from the second-best method, K-means, call into question the accuracy of the classification by the standard technique. Finally, a number of recommendations for improvement of pattern discovery algorithms and discretization techniques are made.

# ACKNOWLEDGEMENTS

I would like to gratefully acknowledge the enthusiastic supervision and tireless editing of Professor Tony Kusalik during this work. I would also like to thank Professors Mark Daley, John DeCoteau, Eric Neufeld, and Chris Soteros for agreeing to be a part of my supervisory committee. Also, Professors Grant Cheston, Mark Keil, and Mik Bickis have helped greatly in exploration of various avenues in this research. Financial support was received from the department of Computer Science Graduate Teaching fellowship, and Professor John DeCoteau. All computers used for this thesis were bought with assistance from the Canadian Foundation for Innovation.

# TABLE OF CONTENTS


1 INTRODUCTION 1

2 BACKGROUND 6

2.1 Central Dogma 7

2.2 Microarrays 7

2.3 Microarray Data Analysis 10
2.3.1 Hierarchical Clustering with "Cluster" 10
2.3.2 Microarray analysis with pattern discovery 13

2.4 Binning 16
2.4.1 Training 17
2.4.2 Sign-of-the-derivative ('+/-/0'-technique) 17
2.4.3 K-means 17
2.4.4 Pre-set Values 18

2.5 Pattern Discovery Applications 19
2.5.1 Teiresias 20
2.5.2 Splash 24
2.5.3 Genes@Work 27
2.5.4 Comparison 32
2.5.5 Tupleware 33

3 DATA & METHODOLOGY 34

3.1 Data 36
3.1.1 Yeast 37
3.1.2 Lymphoma 38

3.2 Methodology 39
3.2.1 Data Binning and Filtration 40
3.2.2 Aligned and Unaligned Patterns 43
3.2.3 Training Data 44
3.2.4 The Genes@Work (Training-Data) Approximation 48
3.2.5 Yeast Cell-cycle 50
3.2.6 Lymphoma 56
3.2.7 Other Issues 60
3.2.8 Cluster 60

3.3 Materials 61

4 RESULTS 61

4.1 Yeast Cell-cycle 61
4.1.1 Missing Pattern Problem 62
4.1.2 Time and Space Usage 64
4.1.3 Pattern Properties in a subset of the YCC data 70

4.1.4 Sign-of-the-derivative technique 74
4.1.5 Hierarchical Clustering with Cluster 97

**4.2 Lymphoma Results 104**
4.2.1 Clustering with Xmeans 106
4.2.2 Pre-set value binning 118
4.2.3 Genes@Work Approximation 123
4.2.4 Classification with Genes@Work 123

**5 DISCUSSION AND FUTURE WORK 132**

**6 REFERENCES 143**

# LIST OF TABLES


TABLE 3.1. DATA, BINNING TECHNIQUES AND APPLICABLE PROGRAMS....40
TABLE 4.1. EXECUTION TIME AND MEMORY USAGE FOR TEIRESIAS, SPLASH AND GENES@WORK....65
TABLE 4.2. NUMBER OF PATTERNS DISCOVERED BY TEIRESIAS, SPLASH AND GENES@WORK....66
TABLE 4.3. NUMBER OF ALIGNED PATTERNS OF GIVEN LENGTH FOR VARYING SUPPORT....72
TABLE 4.4. THE EXECUTION TIME AND MEMORY USAGE OF TEIRESIAS AND GENES@WORK FOR EACH VALUE OF E AND *K* FOR SIGN-OF-THE-DERIVATIVE DATA BINNING TECHNIQUE. ....77
TABLE 4.5. LONGEST PATTERNS DISCOVERED BY GENES@WORK FOR E=0.1, *K*=2....81
TABLE 4.6. LEAST PROBABLE PATTERNS DISCOVERED BY GENES@WORK FOR E=0.1 *K*=2....82
TABLE 4.7. LONGEST PATTERNS DISCOVERED BY GENES@WORK FOR E=0.1 *K*=3....82
TABLE 4.8. LEAST PROBABLE PATTERNS DISCOVERED BY GENES@WORK FOR E=0.1 *K*=3....83
TABLE 4.9. LONGEST PATTERNS DISCOVERED BY TEIRESIAS FOR E=0.1 *K*=6....83
TABLE 4.10. LEAST PROBABLE PATTERNS DISCOVERED BY GENES@WORK FOR E=0.1 *K*=6....84
TABLE 4.11. LONGEST PATTERNS DISCOVERED BY TEIRESIAS FOR E=0.2 *K*=2....85
TABLE 4.12. LEAST PROBABLE PATTERNS DISCOVERED BY GENES@WORK FOR E=0.2 *K*=2....85
TABLE 4.13. LONGEST PATTERNS DISCOVERED BY TEIRESIAS FOR E=0.2 *K*=3....86
TABLE 4.14. LEAST PROBABLE PATTERNS DISCOVERED BY GENES@WORK FOR E=0.2 *K*=3....86
TABLE 4.15. LONGEST PATTERNS DISCOVERED BY TEIRESIAS FOR E=0.2 *K*=6....87
TABLE 4.16. LONGEST (AND LEAST PROBABLE) PATTERNS DISCOVERED BY GENES@WORK FOR E=0.4 *K*=6....87
TABLE 4.17. THE LONGEST PATTERNS DISCOVERED WITH *K*=2....91
TABLE 4.18. THE LONGEST PATTERNS DISCOVERED WITH *K*=3....92
TABLE 4.19. THE LONGEST PATTERNS DISCOVERED WITH *K*=4....93
TABLE 4.20. THE LONGEST PATTERNS DISCOVERED WITH *K*=5....94
TABLE 4.21. THE LONGEST PATTERNS DISCOVERED WITH *K*=6....94
TABLE 4.22. LONGEST PATTERNS OF SUPPORT 2....96
TABLE 4.23. LONGEST PATTERNS OF SUPPORT 3....97
TABLE 4.24. FOR EACH DATA SET THE NUMBER OF GENES THAT HAD MORE THAN ONE BIN, AND THE EXECUTION TIME ON THE BEOWULF CLUSTER IS GIVEN....107
TABLE 4.25. THE PERCENTAGE OF PATTERNS OF SUPPORTED ONLY BY THE ABC SUBTYPE, THE MAXIMUM SUPPORT FOR ABC SUBTYPE PATTERNS, AND THE MAXIMUM SUPPORT FOR GCB SUBTYPE PATTERNS FROM THE TOP 20 HIGHEST SUPPORT PATTERNS....109
TABLE 4.26. FOR EACH DATA SET ($K_{MAX}$=10, 15, 20) THE NUMBER OF TOP 20 PATTERNS WHOSE SUPPORTING TISSUES CONSIST OF 80%, 90% OR 100% ABC SUBTYPE FOR EACH TYPE OF MINIMUM GENE CRITERIA ($G_{MIN}$=2, 3, 4, 8) OR MINIMUM SUPPORT CRITERIA ($S_{MIN}$=5, 10, 14, 16)....111
TABLE 4.27. FOR EACH DATA SET ($K_{MAX}$=10, 20) THE NUMBER OF TOP 20 PATTERNS WHOSE SUPPORTING TISSUES CONSISTS OF 80%, 90% OR 100% GCB SUBTYPE FOR EACH TYPE OF MINIMUM GENE ($G_{MIN}$=2, 3, 4, 8) OR MINIMUM SUPPORT ($S_{MIN}$=5, 10, 14, 16) CRITERIA....113
TABLE 4.28. THE NUMBER OF PATTERNS THAT SUCCESSFULLY CLASSIFY, AT THE GIVEN PERCENTAGE, EITHER THE ABC OR GCB SUBTYPE FOR THE LISTED MINIMUM GENE REQUIREMENT....119
TABLE 4.29. THE NUMBER OF PATTERNS THAT SUCCESSFULLY CLASSIFY, AT THE GIVEN PERCENTAGE, THE ABC AND GCB SUBTYPES FOR THE LISTED MINIMUM SUPPORT REQUIREMENT. ....120
TABLE 4.30. TOP 10 PATTERNS DISCOVERED BY GENES@WORK WHEN ORDERED BY SUPPORT....125
TABLE 4.31. TOP 10 PATTERNS DISCOVERED BY GENES@WORK WHEN ORDERED BY LENGTH....126
TABLE 4.32. THE TOP 10 PATTERNS DISCOVERED BY GENES@WORK WHEN ORDERED BY PROBABILITY..126
TABLE 4.33. CANDIDATE CLASSIFIER GENES FOR ABC SUBTYPE DISCOVERED BY GENES@WORK....128
TABLE 4.34. THE TOP 10 PATTERNS SORTED BY LENGTH, DISCOVERED BY GENES@WORK FOR THE ABC DLBCL SUBTYPE. ....129
TABLE 4.35. THE TOP 10 PATTERNS SORTED BY PROBABILITY, DISCOVERED BY GENES@WORK FOR THE ABC DLBCL SUBTYPE. ....130
TABLE A.1. TOP 20 HIGHEST SUPPORT PATTERNS WITH WITH XMEANS BINNING $K_{MAX}$=10....146
TABLE A.2. TOP 20 HIGHEST SUPPORT PATTERNS WITH WITH XMEANS BINNING $K_{MAX}$=20....147
TABLE A.3. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING $K_{MAX}$=10 (GENE NAMES OMMITTED)....148
TABLE A.4. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING $K_{MAX}$=20 (GENE NAMES OMMITTED)....148

TABLE A.5. TOP 20 HIGHEST SUPPORT PATTERNS WITH XMEANS BINNING, $K_{MAX}$=10, WITH 100% ABC OR GCB SUPPORT ..... 149
TABLE A.6. TOP 20 HIGHEST SUPPORT PATTERNS WITH XMEANS BINNING, $K_{MAX}$=20, WITH 100% ABC OR GCB SUPPORT ..... 150
TABLE A.7. TOP 20 HIGHEST SUPPORT PATTERNS WITH XMEANS BINNING, $K_{MAX}$=10 AND $G_{MIN}$=8 ..... 151
TABLE A.8. TOP 20 HIGHEST SUPPORT PATTERNS WITH XMEANS BINNING, $K_{MAX}$=20 AND $G_{MIN}$=8 ..... 152
TABLE A.9. TOP 20 HIGHEST SUPPORT PATTERNS WITH XMEANS BINNING, $K_{MAX}$=10, ABC<50% AND $G_{MIN}$=8 ..... 153
TABLE A.10. TOP 20 HIGHEST SUPPORT PATTERNS WITH XMEANS BINNING, $K_{MAX}$=20, ABC<50% AND $G_{MIN}$=8 ..... 154
TABLE A.11. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=10, $S_{MIN}$=5 (GENE NAMES OMMITTED) ..... 155
TABLE A.12. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=20, $S_{MIN}$=5 (GENE NAMES OMMITTED) ..... 155
TABLE A.13. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=10, $S_{MIN}$=5, ABC<50% (GENE NAMES OMMITTED) ..... 156
TABLE A.14. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=20, $S_{MIN}$=5, ABC<50% ..... 156
TABLE A.15. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=10, $S_{MIN}$=10 ..... 157
TABLE A.16. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=20, $S_{MIN}$=10 ..... 158
TABLE A.17. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=10, $S_{MIN}$=10, ABC<50% ..... 159
TABLE A.18. TOP 20 LONGEST PATTERNS WITH XMEANS BINNING, $K_{MAX}$=20, $S_{MIN}$=10, ABC<50% ..... 160
TABLE A.19. TOP 20 HIGHEST SUPPORT PATTERNS WITH PRE-SET VALUE BINNING ..... 161
TABLE A.20. TOP 20 LONGEST PATTERNS WITH PRE-SET VALUE BINNING (GENE NAMES OMMITTED) ..... 162
TABLE A.21. TOP 20 HIGHEST SUPPORT PATTERNS WITH PRE-SET VALUE BINNING, WITH 100% ABC OR GCB SUPPORT ..... 163
TABLE A.22. TOP 20 HIGHEST SUPPORT PATTERNS WITH PRE-SET VALUE BINNING, AND $G_{MIN}$=8 ..... 164
TABLE A.23. TOP 20 HIGHEST SUPPORT PATTERNS WITH PRE-SET VALUE BINNING, AND $G_{MIN}$=12 ..... 165
TABLE A.24. TOP 20 LONGEST PATTERNS WITH PRE-SET VALUE BINNING (GENE NAMES OMMITTED), $S_{MIN}$=5 ..... 166
TABLE A.25. TOP 20 LONGEST PATTERNS WITH PRE-SET VALUE BINNING (GENE NAMES OMMITTED), $S_{MIN}$=5, ABC>50% ..... 166
TABLE A.26. TOP 20 LONGEST PATTERNS WITH PRE-SET VALUE BINNING, $S_{MIN}$=7 ..... 167
TABLE A.27. TOP 20 LONGEST PATTERNS WITH PRE-SET VALUE BINNING, $S_{MIN}$=7, ABC>50% ..... 168

# LIST OF FIGURES


FIGURE 2.1. THE DENDROGRAM OF A HIERARCHICAL CLUSTERING TREE. 11
FIGURE 2.2. SCHEMATIC EXPRESSION OF 3 GENES ACROSS 5 CONDITIONS TO SHOW DIFFERENT TYPES OF CORRELATION. 12
FIGURE 2.3. THE EXPRESSION VALUES OF 8 GENES ACROSS 9 CONDITIONS ILLUSTRATING A CLOSE RELATIONSHIP AT 5 POINTS AND A HIGHLY DIVERGENT RELATIONSHIP AT 4 POINTS. 13
FIGURE 2.4. THREE DATA STREAMS WITH ONE ALIGNED AND ONE UNALIGNED PATTERN. 14
FIGURE 2.5. THE EFFECT OF 4 DIFFERENT BINNING TECHNIQUES ON A SINGLE DATA SET. 19
FIGURE 2.6. THE FOUR "WINDOWS" OF LENGTH $W$=4 FROM INPUT $S$. 20
FIGURE 2.7. THE 4 ELEMENTARY PATTERNS FOR "DEFG" WITH $L$=3. 20
FIGURE 2.8. TWO PATTERNS ILLUSTRATING PREFIX-WISE LESS ORDERING. 21
FIGURE 2.9. TWO PATTERNS ILLUSTRATING SUFFIX-WISE LESS ORDERING. 21
FIGURE 2.10. THREE ELEMENTARY PATTERNS FROM SOME THEORETICAL INPUT STREAM. 22
FIGURE 2.11. THE CONVOLUTION OF *EP1* AND *EP2* TO FORM *PNEW1* WITH THE LOSS OF EP2. 22
FIGURE 2.12. THE CONVOLUTION OF *EP1* AND *EP3* FORM THE SUPER PATTERN *PSUP*. 22
FIGURE 2.13. PATTERN P1 SHARED BETWEEN STREAMS 1 & 2 IS NEITHER LEFT- NOR COMPOSITION-MAXIMAL. 25
FIGURE 2.14. AN ALIGNED, SIMILAR PATTERN IS SHARED BETWEEN STREAMS 1 & 2. 26
FIGURE 2.15. DATA STREAMS 1 & 2 ARE BINNED INTO 1' AND 2' RESPECTIVELY. PATTERN 1 FORMS WITHOUT EQUIVALENCE CLASSES, PATTERN 1' FORMS WHEN ADJACENT BIN EQUIVALENCE CLASSES ARE USED. 27
FIGURE 2.16. A PROBABILITY DENSITY DISTRIBUTION $P$ FOR SOME GENE $U$, WITH THE AMOUNT OF EXPRESSION ON THE X-AXIS AND ITS PROBABILITY ON THE Y-AXIS. THREE PARTICULAR EXPRESSION AMOUNTS ARE SHOWN, U1, U2 AND U3. 30
FIGURE 2.17. THE SAME PROBABILITY DENSITY DISTRIBUTION PICTURED ABOVE AFTER TRANSFORMATION BY GENES@WORK. 30
FIGURE 3.1. A FLOW CHART OF THE METHODOLOGY TO BE EMPLOYED. 35
FIGURE 3.2. THREE STREAMS BEFORE AND AFTER FOREIGN CHARACTER INSERTION TO DEMONSTRATE THE BREAKING-UP OF UNALIGNED PATTERNS. 44
FIGURE 3.3. TWO GENES WITH VERY DIFFERENT EXPRESSION PROFILES UNDER SOME EXPERIMENTAL CONDITION. 45
FIGURE 3.4. GENE EXPRESSION VALUES FOR TWO GENES FROM THE PREVIOUS FIGURE IN THEIR BASE STATE. 46
FIGURE 3.5. PROBABILITY DENSITY DISTRIBUTION OF A GENE'S EXPRESSION VALUES WITH THREE EXPRESSION LEVELS INDICATED. 46
FIGURE 3.6. EXPERIMENTAL AND TRAINING GENE EXPRESSION DATA 47
FIGURE 3.7. RANGES FOR TRAINING DATA WHEN SUED TO CREATE 4 BINS. 50
FIGURE 3.8. TEST DATA AND RESULTING CHARACTERS USING THE TRAINING DATA AND BIN RANGES FROM PREVIOUS FIGURE. 50
FIGURE 3.9. THE EFFECTS OF DIFFERENT VALUES FOR $K$, THE NUMBER OF CENTROIDS 52
FIGURE 3.10. K-MEANS BINNING DATA BEFORE AND AFTER MERGING INTO 3 BINS 53
FIGURE 4.1 LOG EXECUTION TIME OF TEIRESIAS, SPLASH AND GENES@WORK VS. NUMBER OF STREAMS (WITH CONSTANT SUPPORT, $K$=2) , WITH LINEAR INTERPOLATION OF THE DATA POINTS 67
FIGURE 4.2. MEMORY USED VS. THE NUMBER OF PATTERNS DISCOVERED FOR TEIRESIAS, SPLASH AND GENES@WORK FOR 25, 30, 35 AND 40 DATA STREAMS AND $K$=2…5, WITH LINEAR INTERPOLATION OF THE DATA POINTS 69
FIGURE 4.3 ALIGNED PATTERNS DISCOVERED BY TEIRESIAS VS. THE NUMBER OF STREAMS IN THE DATA SET TO SHOW EXPONENTIAL RELATIONSHIP, SHOWN AS A SOLID LINE 73
FIGURE 4.4. HIERARCHICAL CLUSTERING OF DLBCL SUBTYPES BY ALIZADEH, WITH ADDED NUMBERING OF TISSUE SUBTYPES. 105

# 1 Introduction

A living organism is an intricate set of chemical reactions, complex compounds and remarkably ordered structures. The delicate balance that is staying alive – consuming energy, producing metabolites, maintaining physiological integrity in a chaotic environment and ultimately, reproducing – is mediated by an organism's genes. From a simple yeast with seven-thousand genes to a complex human with over twenty-thousand genes, the ability of an organism to survive, adapt and grow is mediated by the environment's interaction with its genes and the interaction of its genes with one another.

Thanks to large-scale sequencing projects and gene prediction algorithms, the sequences of every gene in a number of organisms is known. However, and perhaps counter-intuitively, the function of all the genes is not. An important goal in molecular biology is to elucidate the function of every gene in an organism, or more generally, to discover genes that are important in certain biological roles, like the reaction to a stress such as heat shock, the progression of a disease, for example, cancer, or the function of an organ, like a liver or kidney.

In a very general way, the mechanism of every gene is known. Succinctly, each gene encodes a product that is important to some function in a cell. The amount of the produced is the gene's *expression*. As the requirements of an organism change so too does the expression of each gene. Therefore, by monitoring the expression of a gene under some condition we can determine its importance to the function of the cell in that state. More generally, genes that react similarly under a condition or set of conditions likely participate in the same (or a similar) biological role. For example, if a group of genes are highly co-expressed in Cancer A and not highly expressed in Cancer B we might conclude that those genes are important in the progression, function or mediation of Cancer A. We can also look at this problem from a slightly different angle: we can consider this group of genes to define Cancer A, and use them to differentiate (or classify) A from B.

The problem then arises of how to measure the expression of not just a single gene, but of every gene in an organism. Luckily, a system called a gene expression microarray is capable of measuring any gene's expression provided that the gene's sequence is known. Further, microarrays are capable of measuring tens of thousands of genes in a single experiment. Microarrays that can measure the expression of all (or nearly all) of the genes in humans, yeast and a number of other organisms are currently available. Because each microarray represents a single condition, disease state or tissue sample, it is common to refer to each experiment type as simply an array or chip.

Unfortunately, there are four major difficulties encountered when measuring gene expression with microarrays. Firstly, a gene's expression is stochastically variable. That is, some gene that is qualitatively considered to be up-regulated (it produces more mRNA) may be measured in a quantitative fashion that fluctuates around some true value, for instance, as any of a 2.0-, 2.2-, or 3.0-fold increase. Secondly, gene interactions are complex but the observable effects on an organism are simple. A single observable state, such as a disease, may be mediated by different genes in different situations. Thirdly, microarrays are a noisy measuring tool. That is, the variability of a gene, even one that is highly consistent in its expression, may be high [Baldi, '02]. The third problem is partially due to the many (unautomated) steps involved in a microarray experiment which tend to build up the amount of noise. Interestingly, each of these problems could be (at least partially) resolved by performing a large number of replicate experiments. This, however, highlights the fourth problem encountered when using microarrays: they are expensive to perform. Despite these inherent problems, microarray technology is extensively used by researchers and is considered a key advancement in molecular biology.

One problem molecular biologists have not been able to contend with is the enormous amount of data that is produced from a microarray experiment and the combinatorial nature of the analysis that is required to find relationships between the thousands of genes that compose an organism. This problem has motivated the use of computers and data analysis techniques to analyze microarray data. Generally, the analysis seeks to

group arrays and/or genes together in some biologically significant manner. Artificial neural networks (ANNs), support vector machines (SVMs) and similar techniques have been applied to disease classification [Brazma, '00]. Hierarchical clustering, k-means, and self-organizing maps (SOMs) have been used to cluster both genes and arrays into meaningful groups [Brazma, '00].

Pattern discovery is another technique that can be applied to creating gene/array groups. It is an advantageous analysis technique because it clearly indicates which genes are important in defining a cluster of arrays and/or the arrays that are important in defining a cluster of genes. This is in contrast to the above techniques which may make it difficult to determine which genes/arrays are important for defining the group.

Pattern discovery is not without its own concerns. Firstly, pattern discovery is a hard problem. Searching all combinations of genes and conditions is impractical and many pattern discovery algorithms use heuristics (an algorithm that yields non-optimal results but provides a decrease in execution time) to trim the problem space. Secondly, most pattern discovery techniques require the data to be composed of values from a discrete domain, rather than numbers from a continuous[1] universe such as is found in microarray data. Thus some method of discretizing the data is usually required.

Teiresias [Rigoutsos, '98], Splash [Califano, '99] and Genes@Work [Califano, '00] are three pattern discovery applications. Teiresias and Splash use slightly different methods of discovering patterns in character data and likely differ in their speed and memory efficiency. Interestingly, these two applications (and their underlying algorithms) have never been benchmarked against one another in the open literature. Genes@Work was born from the work done on Splash and is designed specifically for use with microarray data. Unlike the other two applications Genes@Work can discover patterns in continuously valued (too finely grained) data.

[1] Technically, microarray data consists of "finely grained" discrete data. However, in this work, "continuous" will be used to refer to data that is too finely grained for pattern discovery.

Generally, this thesis seeks to explore the use of patterns for gene microarray analysis in a breadth-first manner; rather than fully explore a single aspect of analysis it will touch on every step involved in analysis. Specifically, this work has two goals. First, it seeks to discover which of the three pattern discovery algorithms are the most efficient in terms of execution time (as measured by elapsed time) and memory efficiency for particular microarray data analysis problems. Second, it endeavors to determine which of 4 methods of discretization results in the most interesting patterns being discovered. This is accomplished using two microarray data sets. The first is typical of data found when relationships between genes are to be discovered, and the second when associations between the arrays must be discovered. In order to achieve the second goal, two other aspects of gene microarray analysis using pattern discovery will be explored. First, ways in which to pick patterns, based on their length and support, so as to favor biologically significant patterns over random ones, will be investigated and compared to Genes@Work's more complicated ranking system which relies on measuring the probability of a pattern's occurrence. Second, specifically for the second data set, a simplistic system to classify arrays using patterns will be developed (see Sections 3.1.2 & 3.2.6) and from these patterns a set of genes that appear to be good classifiers will be chosen.

As indicated above, this thesis has broad reaching goals. That is, it covers a number of areas of microarray analysis and pattern discovery, but none in an exhaustive fashion. The work is conducted in such a breadth-first manner for two major reasons. Firstly, little work has been done using pattern discovery for microarray analysis. By working with multiple data sets, applications, and binning techniques the prospect of finding a good technique is increased. Secondly, it is difficult to judge the quality of the discovered patterns without a good way of selecting/clustering them. This allows good patterns to be selected from random ones. However, it is difficult to judge the quality of a selection/clustering method unless it is known that good patterns are being discovered by the application. Therefore, both aspects need to be explored at some level in order to judge the biological significance of any binning technique in a fair manner.

This thesis consists of four major sections, the Background, the Data and Methodology, the Results and the Conclusions and Future Work. The Background consists of five main subsections. First the fundamental dogma of molecular genetics is explained in greater detail. This is followed by Section 2.2 where microarrays, microarray experiments and the representation of microarray data are detailed. In Section 2.3 various microarray data analysis techniques are quickly explored which leads into a discussion of what is meant by a pattern and what pattern discovery is. The Background then covers different discretization techniques, which in this work is termed *binning*. Finally, Section 2.5 explains the three different pattern discovery applications and concludes with a means of comparing them.

The Data and Methodology section begins with a general overview of the techniques employed for this work. Section 3.1 outlines the two data sources used. Sections 3.1.1 and 3.1.2 discuss data preprocessing. Section 3.2.3 outlines what training data is and why it is required. This leads into Section 3.2.4 which covers an approximation for one of the data binning techniques. Then, taking each data set in turn, the exact methodology – from preprocessing to binning to pattern discovery and pattern analysis – is covered in Sections 3.2.5 and 3.2.6. This section concludes with a discussion of a few minor issues.

The results section consists of two major subsections, one for the yeast cell-cycle data and one for the lymphoma data. Any problems that were encountered in the methodology and the solutions to these problems are listed there along with the experimental results for timing, memory usage and binning techniques, as required. Lastly, this work finishes with some general conclusions about the applications, binning techniques, and usefulness of pattern discovery on each of the two data types, along with future work for the field.

# 2 Background

New technologies have enabled the scientific community to sequence the entire genome of an organism (that is, it is possible for scientists to know of every genetic factor involved in growth and development, reactions to disease, and response to changes in environment for an entire species). Unfortunately, this new source of information provides more questions than answers; although the chemical composition of a gene (its sequence) and where it is located in the genome is known, it is not necessarily known what the gene does or how it is regulated. Gene expression microarrays are a technology that attempts to answer these questions. By monitoring all the genes in an organism, microarrays can serve as a basis for relating genes to one another. However, microarray technology has its own problems: for any organism, thousands or tens of thousands of genes exist and must be monitored. Further, most experiments require multiple microarrays to be used. This leads to a large influx of information. The problem then is to reduce and organize the data and ultimately to form groups of related genes; this is broadly termed *microarray analysis*.

As might be expected there are many ways to analyze microarray data. The general ideology is to take raw information and to distill from it knowledge and understanding. Indeed, many of the techniques used in microarray analysis originate from classical computer science methodologies dealing with high dimensional data and data clustering (data mining). This research explores the use of pattern discovery for microarray analysis. It concentrates on two aspects of using pattern discovery on microarrays: the choice of the pattern-finding application and the manipulation of the input data in order to yield the greatest number of meaningful patterns. However, before any in-depth discussion of this can take place, a general understanding of what microarrays are, how they work and the data they produce is needed. Also, comprehension of what patterns are and the algorithms and heuristics used to discover them will be covered. This discussion begins with an outline of the central dogma, the basis of all molecular biology.

## 2.1 Central Dogma

A fundamental tenant of all biology is that our genes make us what we are. It is easy to explain the differences between two organisms, such as a bacterium and a human (or even differences between two humans) by saying that they have different genes. However, while every cell in a human body contains the exact same genes, tissues in this body are very distinct from one another. This differentiation is made possible, in part, by different levels of expression of those genes. Indeed, the expression of genes in a tissue can change over time. After eating a meal, for instance, genes are activated in response to the nutrients consumed and a cascade of genes are stimulated in order to absorb and transport those nutrients to the rest of the body.

The central dogma of molecular biology states that every gene is first *transcribed* (copied) into an intermediate form, mRNA, which is then *translated* into a functional protein. Gene expression is a term that describes the amount of transcription taking place in a cell. Expression levels can vary from cell type to cell type (a kidney compared to a liver), in a single cell type over time (a hair follicle going from normal to producing grey hair), and in a single cell type in different physiological conditions (a skin cell being exposed to sunlight and tanning or to darkness).

Throughout this paper the term "condition" will be taken to mean any of the above three causes for variation in expression levels.

## 2.2 Microarrays

Gene expression microarrays are a recent technology that allows the measurement of the expression of thousands of genes to be conducted in parallel. Microarrays effectively enable a biologist to get a "snap shot" of mRNA production from most or all of the genes in a population of cells at a given moment under a particular set of environmental conditions [Brazma, '00]. For cDNA[2] microarrays gene expression is denoted as a floating-point number that is a ratio of the expression of the gene under a particular

[2] Complementary DNA is DNA that is made from RNA via a reverse transcriptase.

condition versus the expression under some predefined "standard" state. This ratio can be intuitively interpreted as "how many times more/less is the expression of this gene under this condition compared to its 'normal' (or some other) state". This is done because the inherent variability in this type of experimentation is quite high. By taking the expression as a relative indicator of expression one helps to "standardize" the measurements across multiple experiments and from gene to gene on a single chip. Further, this ratio is usually expressed in $\log_2$ form. This is done for two reasons. First, it makes comparing increases and decreases in expression more symmetric: a two-fold increase in expression becomes "1" and a ½-times decrease in expression becomes "-1", independent of the absolute measures involved. Second, it has been shown that expression levels, measured multiple times, appear to come from a log-Gaussian distribution [Baldi, '02]. For more information on microarray construction, types, and experimental protocols see the books by Draghici [Draghici, '03] and Baldi [Baldi, '02].

Generally, for any microarray experiment many microarray "snap shots" are taken of a population of cells, each under a different condition, and together they form the raw data used in analysis. The whole of this data can be visualized as a matrix, wherein the rows represent the expression of a single gene under various conditions (called the gene's *expression profile*), and the columns represent the expression of all the genes under a specific condition (a single experiment/array/treatment). The general goal of microarray analysis is to form subgroups of the rows (or columns) of the data matrix that are related in terms of expression. These subgroups are useful for identifying novel relationships between genes (or tissues), providing information on the role of a gene by relating it to genes of known function, and discovering unsuspected regulation networks. The underlying assumption of all microarray interpretation is that genes that are related in terms of expression are related in function, too.

There are three main categories of microarray data. The first, time-series data, is information repeatedly collected from a single population of cells over a period of time. Importantly, the columns in these data have order; that is, one time logically follows another. It intuitively follows that permuting the columns of the data matrix makes little

sense from an analysis standpoint. An example of this type of data would be collecting samples from a synchronized, growing population of yeast cells to monitor gene expression over the cell-cycle [Spellman, '98]. In the second kind of microarray experiment a single type of cell is exposed to different environmental conditions (e.g. heat shock, the addition of hormones, irradiation, etc.)[Gash, '01]. In this case the data columns have no order; any experiment could occupy any column of the data matrix and it would still make organizational sense. The third type of data is where different, possibly related, populations of cells are collected and sampled (usually against some common "normal" population). Depending on the question to be answered, this type of data could include different tissues from a single individual, the same tissue from multiple individuals, or related tissues under a common condition. An experiment where different types of related cancers are tested would be an example of the third type of microarray experiment [Alizadeh, '00].

For any microarray data set (matrix) two types of analysis can occur: comparison of the rows of the matrix to find similarity (relationships between genes), or comparison of the columns of the matrix to find similarity (relationships between microarrays). When a researcher wishes to know "Which of these genes act similarly?" the researcher conducts analysis of the first type. When a researcher seeks to find relationships between the columns of the matrix, the researcher is attempting to answer: "Which times/conditions/tissues are related?". Occasionally analysis in both directions is pursued when the relatedness only holds/exists for a subset of all the genes or trials [Kluger, '03]. Generally, for the first two types of microarray data, one compares genes (rows) to one another, and in the third type of data, one compares tissue samples (columns).

Relatively speaking, the number of columns in microarray data is small, consisting of tens of experiments. However, the number of rows in a data set can be quite large: ~6,000 for a yeast experiment, and up to ~20,000 or more for human experiments. Often, because of the expense involved in constructing a full set of microarray experiment data, no replicate experiments are conducted. Some microarrays have replicate spots on a single chip (which allows repeat observations of gene expression for an experiment).

However, these chips are less common as the need to have more genes on a single chip increases. Also, there may be some (and occasionally many) missing data points in the matrix. Some genes seem especially prone to giving poor results: any "signal" present is consistently hidden by background noise. Because of this, a data matrix often has rows that are missing a large proportion of their data points.

## *2.3 Microarray Data Analysis*

Once the data has been collected it must then be analyzed. Many different forms of analysis have been used to examine microarray data. Due to the high dimensionality of the data (each gene/tissue is composed of many data elements), Self-Organizing Maps (SOMs) and Principal Component Analysis (PCA) are obvious approaches [Baldi, '02]. The complexity inherent in microarray data has prompted the application of Bayesian networks and other small-sample statistical models for the examination of microarray data [Baldi, '02]. Also, more traditional clustering techniques (K-means) have been used [Baldi, '02].

In particular there are two analysis techniques of special interest: hierarchical clustering ("HC") [Eisen, '98], and pattern discovery [Rigoutsos, '00]. HC was one of the first analysis techniques to gain wide-spread popularity and use and is the "gold standard" to which other methods are compared. Pattern discovery is a novel technique which offers several advantages over the more traditional approaches and is the focus of this research.

### 2.3.1 Hierarchical Clustering with "Cluster"

The "Cluster" application is a hierarchical clustering application that is the standard method for microarray data analysis by most biologists. It was developed for microarray analysis and was used in one of the first and most cited papers involving microarrays [Eisen, '98]. "Cluster" does have several short comings, but despite this, it is still one of the most widely used microarray analysis tools.

"Cluster" treats each input stream (the rows or columns of the data matrix) as a point in high-dimensional space and groups these points using hierarchical clustering. For each iteration of the program a pair-wise distance between every stream (or group of streams) is calculated using some distance metric (Pearson correlation coefficient, Euclidian distance, etc.). The two closest points are then grouped together. For multi-membered groups the distance between them can either be the average distance between all members of the groups (average linkage), the distance between the closest points (single linkage), or the distance between the farthest points (complete linkage). This process continues until all streams are collapsed into a single group. In Figure 2.1, the final group can be seen and thought of as the root of a tree, and extending from it, all of the children. The leaves of this tree are the original input streams [Eisen, '98]. Points that are collapsed earlier in the process (further from the root) are more related than points that are collapsed later.

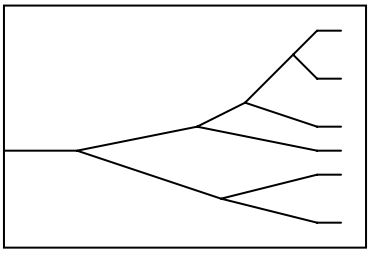

**Figure 2.1. The dendrogram of a hierarchical clustering tree.**

Despite its popularity, the use of "Cluster" has many drawbacks. The first is in the selection of a distance metric on which to base the clustering. As an example in Figure 2.2, we see the expression levels of three genes (*A B* and *C*) across 5 conditions. If the Euclidian distance metric is used genes *B* and *C* are considered close, if the Pearson correlation coefficient is used *A* and *B* are considered close together. Clearly the two methods can yield different results either of which might be capturing a real biological relationship. Generally, a researcher will pick one method over another (somewhat arbitrarily), and will therefore miss one of the relationships.

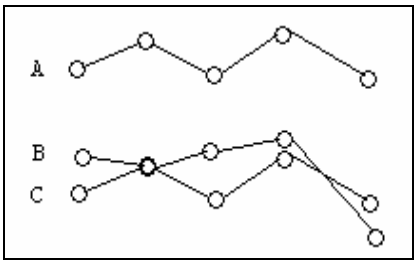


**Figure 2.2. Schematic expression of 3 genes across 5 conditions to show different types of correlation.**

Further, in hierarchical clustering a choice must be made on how to calculate the distance between multi-membered groups (termed linkage); one linkage type may give very different results from another. However, there is no way to know which linkage method better captures real biological relationships, and which method may simply capture coincidental relationships. Often a researcher may simply pick a method that yields the results he wishes to find, rather than maximizes some mathematical properties of the clusters (such as separation between clusters, or conformity within clusters). This may lead to erroneous results. For further information on hierarchical clustering and other techniques see the review paper by Brazma [Brazma, '00] and the book by Draghici [Draghici, '03].

Lastly, "Cluster" (and most other clustering techniques) can only show binary relationships. That is, all distances are only calculated between two members of the input set. However, we should expect that clusters should contain many members and only finding pair-wise relationships between members may not be enough to form these clusters. In Figure 2.3 we see 8 genes that share a few data points in common. Taken together this relationship is significant. However, when only considering any two of the streams at one time the relationship may not be considered significant and thus the genes may be placed in other groups and this may cause grouping to never form.

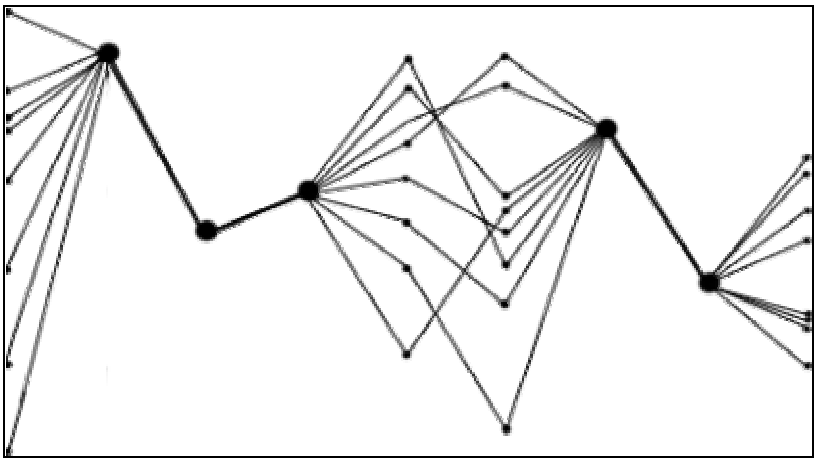

**Figure 2.3. The expression values of 8 genes across 9 conditions illustrating a close relationship at 5 points and a highly divergent relationship at 4 points.**

Despite its shortcomings, hierarchical clustering has become the most common form of microarray analysis[3] (it is available in virtually all commercial and public microarray analysis packages), and because of this it will be used to analyze the three types of data presented in the Data section. As such it will serve to establish a basis of comparison for the other analysis. That is, novel results can be confirmed or disputed by using the results from hierarchical clustering.

### 2.3.2 Microarray analysis with pattern discovery

Pattern discovery is another form of analysis that has been used on gene expression microarray data [Rigoutsos, '00. Califano, '00]. Although this technique has met with some success, it is not very popular. However, before pattern discovery can be discussed in detail what is meant by a "pattern" must first be defined.

[3] Most commonly Pearson's r and average-linkage is used for analysis with Cluster, although other distance metrics and linkages are available.

### 2.3.2.1 Patterns

A *pattern* is an ordered set of one or more characters from some finite alphabet Σ (a finite set of symbols), shared between multiple strings (henceforth called *streams*) and is considered *interesting* [Rigoutsos, 00]. The definition of "interesting" is application specific: long or frequent patterns are commonly considered interesting, but patterns with certain characters or with characters in certain positions may be interesting in some cases. A pattern is specified by a *template* and a list of occurrences termed the *offset list*. The template is a series of *literals* optionally interspersed with a "don't care" character (the '.' character). Each item of the list of occurrences records a stream identifier and the position (*offset*) of occurrence. No template will begin or end with a '.' because the character provides no new information about the pattern; it simply changes values in the offset list. The cardinality of the list is the *support* for the pattern. Patterns that occur with the same offset in all streams are considered *aligned*. If the offsets differ by some finite amount the patterns are *unaligned*. In Figure 2.4, the patterns appear below the streams. The characters within double-quote marks is the template. The co-ordinate set following the template is the offset list. The unaligned pattern P1 occurs in streams 1 and 2 at an offset of 2 and 3, respectively, and the aligned pattern P2 occurs in Streams 2 and 3 at an offset of 0.

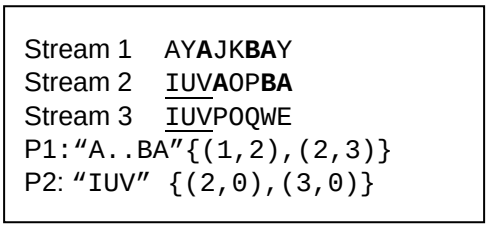


**Figure 2.4. Three data streams with one aligned and one unaligned pattern.**

It is important to note that unaligned patterns only make sense in sequential data: data in which permuting the order of the events is not allowed because its order is set. An example of this type of data is time series data, because the point "10 minutes" logically follows "5 minutes" and precedes "15 minutes". An unaligned pattern from this type of data indicates that some set of genes share a pattern displaced in time. In Figure 2.4, if we imagine the data that comprises the streams is collected in 5-minute intervals, then the pattern P1 occurs in both Streams 1 and 2 but it occurs in Stream 2 five minutes after it

occurs in Stream 1. An example of data where there is no order to the events in a stream is data collected under some environmental conditions *A*, *B* and *C*. This data is considered unordered because it does not matter, generally, whether the data from condition A is listed before the data from condition *B* or *vice versa*. In Figure 2.4, if we imagine the data that comprises the streams is from a set of environmental conditions then pattern P1 indicates that Streams 1 and 2 happen by chance to react similarly in unrelated environmental conditions.

#### 2.3.2.2 Pattern Discovery

Pattern discovery is an advantageous microarray analysis technique because it finds local areas of similarity in each gene's expression profile whereas HC can only measure distance using the entire gene's expression profile. In HC, strong local areas of similarity become "diluted" when the relationship between two streams is considered "globally" thus masking a strong local similarity even when this relationship may be echoed by other genes in the data set. This situation is just another way of thinking of the example in Figure 2.3. In this figure, the expression of the 8 genes are exactly the same across 5 different points, however, there may be suitable differences across the other time points to mask this similarity. A "real world" example of this can be found in "The Emergence of Pattern Discovery Techniques in Computational Biology" [Rigoutsos, '00] where 17 genes from a yeast cell cycle data set have the exact same expression across 17 time points, but are suitably divergent in the remainder of the data set to mask the presence of the relationship when using HC with the Pearson correlation coefficient analysis technique.

Pattern discovery also allows the discovery of patterns that occur "out of phase"; that is, patterns that are displaced across treatments (i.e. displaced in time). This type of pattern may occur in time-series microarray data. Lastly, pattern discovery can intelligently "ignore" missing data points because pattern discovery does not require that every data stream have all data points whereas the Pearson correlation coefficient and many other distance metrics do. This frees the researcher from having to implement a scheme to replace the missing data with a value.

Despite its advantages, pattern discovery has several drawbacks. The first is that patterns (as defined here) can only be found in discrete data. As stated above, raw microarray data consists of floating-point numbers. Therefore, the first step in using pattern discovery to analyze microarray data is to stratify the floating-point data (which is continuous) into a discrete, finite alphabet. In the context of this work, the alphabet is the set (or a subset) of the printable ASCII characters and the process is termed *binning*. ASCII characters are used because many pattern finding applications were developed for motif finding in proteins - which use the characters A-Z to represent amino acids – or arbitrary text. Also, ASCII characters typically require less memory to store than numerical values. The second drawback is that pattern discovery is an NP-Hard[4] problem [Brazma, '98]. For even a small data set the total number of discovered patterns can be huge and the potential number of patterns grows exponentially with input [Rigoutsos, '98]. Because of this, real world application of pattern discovery requires the use of heuristics. Initial work has shown that even using heuristics on a relatively small subset (24x80) of a complete microarray data matrix (6000x80), pattern discovery can take a long time to finish. Also, because many of the patterns must be contained in memory (occasionally a few can be written to disk before all patterns have been found because it is possible to determine the pattern is complete), the memory requirements of pattern discovery are sizable. For example, using a modest data set (620x80), and modest input parameters to Teiresias (L=5, W=9, K=8) (see Section 2.5.1) it is possible to use more than 4GB of memory.

## 2.4 Binning

Binning floating-point numbers into ASCII characters is the first step in microarray pattern analysis. Determining the number of bins and the ranges of values that fall into each bin is not a straight-forward task. Three techniques, however, have shown promise.

[4] NP-Hard problems are intrinsically harder than NP-complete problems (problems that can be solved by a nondeterministic Turing machine in polynomial time), and include the optimizations of NP-complete problems [Brassard, '96].

Although each method varies in how it bins the data, all methods are very quick and usually only take a few seconds to complete.

### 2.4.1 Training

The first method, implemented originally by Genes@Work (see Section 2.5.3), involves the use of two sets of data: the *test* data and a *training set* [Stolovitzky, '03]. Each row (or column) of the training set is uniformly binned into *B* different bins; that is, each bin has the same number of data points in it. The ranges of values that correspond to each of the *B* bins are then applied to the "test" data. This has the effect of inducing a fine granularity in ranges where the training set has many values, and a coarse granularity in ranges where the training set has few values (See Sections 3.2.3 & 3.2.4). What the training data "teaches" the algorithm, is the normal/expected expression of a gene.

### 2.4.2 Sign-of-the-derivative ('+/-/0'-technique)

The second technique is best applied to time-series data [Rigoutsos, '00]. It involves mapping the real-valued expression levels to 3 values: '+', '-' and '0'. The character '+' is used for time point $T_{t+1}$ if $(T_{t+1} - T_t) > +\varepsilon$, '-' if $(T_{t+1} - T_t) < -\varepsilon$; and '0' otherwise, where epsilon is some arbitrarily chosen small number.

### 2.4.3 K-means

The third technique involves the use of K-means clustering. The goal is to find "K number of means" that the input data (a single row or column from the data matrix) seems to cluster into. Using some distance metric $D(X, Y)$ for a set of values **V,** the K-means algorithm can be implemented thusly:

1: initialize each of the $K$ means to $C_j$ ($j$=1 … $K$)
2: For each value $V_i$ in **V,** place it in cluster j if $D(V_i,C_j)$ is minimized
3: For each cluster $j$, set $C_j$ to the mean of all points in cluster $j$
4: repeat steps 2 and 3 until each value $Cj$ converges or after some large number of iterations

For more information see "Clustering Algorithms" by Hartigan ['75].

Two questions arise when using the K-means algorithm. The first is what value of K to use. The second is how to initialize the values $C_j$. A solution to the second problem is to initialize each $C_j$ to a uniform-randomly[5] chosen value. The second problem is much more difficult, in general, to solve. We shall assume that $2<=K<=|\mathbf{V}|/2<=N$, where $N$ is in the order of 100 when clustering together microarray chips and 10,000 when clustering together genes. Further, the data we wish to cluster is one dimensional, and with the bound on $K$, it is trivial to find all $K$ clusters for all values of $K$. Then for each value of $K$, the "goodness of fit" of the clusters must be determined, while at the same time determining a cost for increasing the value of $K$. This is needed because, in the trivial case where $K$= (the number of unique values in set **V**) all the clusters are perfect (and contain one unique value); yet no actual clustering has taken place.

Xmeans is a program developed to solve the problem of picking $K$ [Pelleg, '03a]. Given some minimum and maximum value for $K$, for instance 1 and $|\mathbf{V}|/2$, Xmeans will iteratively apply K-means clustering to the input data with increasing values of $K$. At each iteration Xmeans evaluates the "goodness" of the clusters using Schwarz's Bayesian Information Criterion (BIC) [Wasserman, '03]. After all values for $K$ have been attempted, the $K$ that gives the best BIC score is reported. Xmeans is available for free for non-commercial use [Pelleg, '03b].

### 2.4.4 Pre-set Values

In this technique the expression of every gene is stratified based on 4 values {-1.00, -0.25,+0.25,+1.00}. These delimiters are chosen to signify transitions between "large decrease", "small decrease", "no change", "small increase", and "large increase" in expression, respectively. Unlike the other techniques presented here, this technique is not gene-specific (relative). That is, similar gene expression values will always be binned

[5] A number from the uniform distribution: $P(x) = 1/(b-a)$; a<=x<=b. Where a is in minimum value for the distribution and b is the maximum value.

similarly regardless of other factors; whereas in the other techniques presented here the expression value of the previous state (such as in the '+/-/0' technique), the values in the training data (such as with Genes@Work), or the other expression values for the gene (such as with K-means) will determine how values are binned. This means that similar values occurring in different gene's expression profiles may get mapped to different bins. Figure 2.5 shows how each of the techniques maps the same data stream to different values. Given the training data {-2.2, -2.2, -2.2 …. -2.2} (see Sections 3.2.3 & 3.2.4) the training data technique separates the data into 2 groups around the number -2.2. The '+/-/0' technique is radically different from any of the others. K-means separates the data into 2 groups (which may be thought of as "2ish" and "5ish") and the Pre-set values technique sees only a single group (very down regulated).

| Data: | -5.4 | -5.1 | -5.2 | -2.0 | -2.1 | -2.3 | -2.6 |
|---|---|---|---|---|---|---|---|
| 2.4.1: | A | A | A | B | B | A | A |
| 2.4.2: | | + | 0 | + | 0 | - | - |
| 2.4.3: | A | A | A | B | B | B | B |
| 2.4.4: | A | A | A | A | A | A | A |

**Figure 2.5. The effect of 4 different binning techniques on a single data set.**

## *2.5 Pattern Discovery Applications*

Although the concept of pattern discovery is straight-forward, finding patterns in an efficient and concise manner can be very difficult. The question arises, then, what algorithm (or application) should be used to discover patterns in the microarray data. Fortunately, there are 3 applications widely available for pattern discovery: Teiresias [Rigoutsos, '98], Splash [Califano, '99], and Genes@Work[Califano, '00]. All three of these were developed at IBM. The first two applications are generalized pattern finding algorithms, designed to find patterns in biological strings (DNA, proteins); the last is an adaptation of Splash specifically for pattern discovery in microarrays. Below, the three applications and their algorithms are introduced and compared. This is followed by a short discussion of another pattern finding program, Tuppleware, which will not be used in this research.

### 2.5.1 Teiresias

Teiresias implements a two-stage pattern-finding algorithm. To avoid searching the entire exponentially sized problem-space involved in pattern discovery, Teiresias makes use of a heuristic that employs 3 parameters (*L*, *W* and *K*) to limit the number of patterns searched for. *L* is the minimum number of literals required to appear in a window of size *W* (*W*>*L*). In stage 1, it uses *L* and *W* to establish a minimum *density* for a given *elementary pattern* (all possible combinations of *L* literals from a given window *W*). Density is the ratio of the number of positions that are occupied by literals over a pattern's length [Rigoutsos, '00]. A high value of *L* relative to *W* will cause high density patterns to be discovered. More precisely, a given input sequence *S* of length $S_{len}$, is split into $S_{len}$-*W*+1 overlapping windows of length *W*. Then for each window of *S*, all ${}_{W}C_{L}$ combinations of *L* literals are chosen from that window to form elementary patterns. Elementary patterns that do not appear at least *K* times in all input sequences (a condition termed *support*) are immediately discarded. The application of these 3 parameters forms a heuristic that can greatly trim the otherwise exponential problem space of pattern discovery. In Figure 2.6, *S*="ABCDEFG" and *W*=4. Thus 4 windows are possible and are shown. In Figure 2.7, all 4 elementary patterns for "DEFG" with *W*=4, and *L*=3 are shown. Note that "DEF" is not "DEF." and "EFG" is not ".EFG" because templates may not begin or end with a '.' (see Section 2.3.2.1).

```
Input: S="ABCDEFG", W=4 →
"ABCD", "BCDE", "CDEF", "DEFG"
```

**Figure 2.6. The four "windows" of length *W*=4 from input *S*.**

```
Input: "DEFG", L=3 →
"DEF", "DE.G", "D.FG", "EFG"
```

**Figure 2.7. The 4 elementary patterns for "DEFG" with *L*=3.**

Each elementary pattern is associated with an offset list that details which sequences (starting at 0), and where in those sequences (the first column being position 0) the pattern appears (see Section 2.3.2.1). After all elementary patterns are enumerated, they

are ordered into two lists, a *suffix-wise* and a *prefix-wise* list. Rather than rigorously define the ordering, it will be illustrated by example. In Figure 2.8, the pattern $P_2$ is prefix-wise less than the pattern $P_1$ because $P_1$'s first '.' character occurs in column 2 (shown by the "‡") whereas $P_2$ has a literal in column 2. However, in 2.9, we see that $P_2$ is suffix-wise more than $P_1$ because $P_2$'s last '.' occurs before $P_1$'s last '.' character (shown by the "‡")when considering the patterns in right-to-left order.

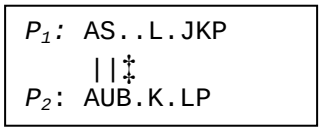


**Figure 2.8. Two patterns illustrating prefix-wise less ordering.**

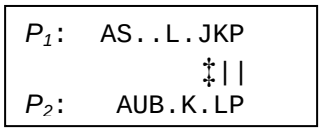


**Figure 2.9. Two patterns illustrating suffix-wise less ordering.**

In the second stage of pattern discovery, Teiresias convolutes patterns which overlap by exactly $L$-1 literals (this can be determined by examining the pattern's template and its offset list) into longer patterns. A new pattern often has a smaller offset list because it will occur less frequently than the shorter patterns that formed it. However, if the offset list of the new pattern is the same size as either one (or both) of the two patterns that formed it then one (or both) of those patterns is removed from the two lists, or rather, subsumed into the new pattern. Teiresias begins the convolution stage with the top-most element on the prefix-wise list and convolutes it with the remaining patterns in the order they occur below it. Each new pattern formed is placed at the top of this list and the process continues until no more convolutions can take place. At that point the pattern is extended using the suffix-wise list in an analogous way. When that is complete the resulting pattern is checked for *maximality* (see below), and if maximal, it is reported. In either event, the pattern is removed from further consideration for convolution. This process is repeated with the next, top most element from the prefix-wise (or suffix-wise) list until both lists are empty. Performing convolution in this way ensures that a *maximal*

(see below) pattern $P_{max}$ is produced before any *superpattern* of $P_{max}$ is produced. A superpattern of $P_{max}$ is any pattern that contains the same (possibly fewer) literals in the same positions as $P_{max}$.

To illustrate the convolution phase of Teiresias Figure 2.10 starts with elementary patterns *EP1*, *EP2* and *EP3* (shown in prefix-wise order). First, *EP1* and *EP2* are convoluted to form $P_{new1}$ and this can be seen in Figure 2.11. Note that $P_{new1}$ has only one member in its offset list and that *EP2* is subsumed by $P_{new1}$ (the cardinality of both $P_{new1}$ and *EP2's* offset lists is the same). Second, $P_{new1}$ is convoluted with *EP3* to form $P_{new2}$. Note that $P_{new1}$ is subsumed into $P_{new2}$ as can be seen in Figure 2.12. At this point $P_{new2}$ cannot be convoluted any further and is checked for maximality (as it is the first pattern to be checked for maximality it cannot be the superpattern of an existing maximal pattern (because there are none) and therefore is maximal). Then *EP1* and *EP3* are convoluted to form a new pattern $P_{sup}$; however this pattern is a superpattern of $P_{new2}$. That is, every literal contained in $P_{sup}$ is already present $P_{new2}$ (at the same positions) and thus is not maximal and would not be reported (and would be removed from the list).

```
EP1: BCD    {(1,5),(2,1)}
EP2: CD..EF {(1,6)}
EP3: A.BC   {(1,3),(2,5)}
```

**Figure 2.10. Three elementary patterns from some theoretical input stream.**

```
EP1: BCD    {(1,5,(2,1)}
EP3: A.BC   {(1,3),(2,5)}
Pnew1: BCD..EF {(1,5)}
```

**Figure 2.11. The convolution of *EP1* and *EP2* to form *Pnew1* with the loss of EP2.**

```
EP1: BCD    {(1,5,(2,1)}
EP3: A.BC   {(1,3),(2,5)}
Pnew2: A.BCD..EF{(1,3)}
Psup:  A.BCD {(1,3)}
```

**Figure 2.12. The convolution of *EP1* and *EP3* form the super pattern *Psup*.**

As *maximal* patterns are discovered they are immediately written to output. A maximal pattern is a pattern that cannot be extended (i.e. cannot have more patterns convoluted onto it) without a decline in the number of members in its offset list and that is not a superpattern of any other discovered pattern. Given, *EP1*, *EP2* and *EP3* are the only elementary patterns in our data set, then *EP1*, *EP2*, and $P_{new2}$ are all maximal because they cannot be convoluted into another pattern without losing support. *EP3* and $P_{new1}$ are not maximal because they can be subsumed (convoluted without losing members of the offset list) into other patterns. $P_{sup}$ is not maximal because it is a superpattern of $P_{new2}$.

The time complexity of Teiresias is linear in the number of patterns it generates as output [Floratos, '98]. However, in the worst case, there are an exponential number of patterns in the output relative to the input. A large difference between *L* and *W*, combined with low values of *L,* allow for very low-density patterns to be discovered. As the density requirement *K* for pattern discovery is lessened more patterns are generated as output. Also, low values of *K* allow infrequent patterns to be kept for output. Both these factors can cause a very large increase in execution time and memory requirements for pattern discovery. In the worst case, in order to find every possible pattern in a data set, *L* is set to 2, *W* to the length of the data stream, and *K* to 2. For all but the smallest and simplest data sets these parameters are unrealistic and pattern discovery is not possible with them. With other values for *K*, *L* and *W* Teiresias guarantees to return all maximal patterns that conform to those parameters.

From previous work it is known that the generation of the elementary patterns is a quick and straight-forward task, usually requiring seconds, or a few minutes to complete and comprising a small fraction of total execution time. The convolution phase, however, can require many minutes, hours or days to complete depending on the input data and parameters.

There is some parallelism to be taken advantage of in the Teiresias algorithm which allows a decrease in execution time. The suffix-wise and prefix-wise lists can be broken up into multiple subsets and thus multiple machines can be tasked with the convolution of

elementary patterns. Also, the act of maximality checking can be spawned as separate processes. This allows the large memory requirements of maximality checking to be broken up across multiple machines.

Teiresias is accessible through a web interface: http://cbcsrv.watson.ibm.com/Tspd.html. However, to access most features Teiresias must be used from the command line. Teiresias has been implemented in the C programming language and is available for AIX, Linux, Sun and Windows operating systems. However, only compiled binaries and not source code are available.

Due to its parallelism Teiresias has also been implemented using MPI to work on a Linux workstation "Beowulf cluster". In such a system the pattern discovery process is broken up across multiple computers. This reduces the memory and computational needs of any single processor and decreases total (wall clock) runtime.

### 2.5.2 Splash

Splash is another pattern discovery application developed at IBM [Califano, '99]. In many respects Splash is highly similar to Teiresias and a number of parallels can be drawn between them. In Splash, the Teiresias parameters for support, $K$, minimum literals, $L$, and windows size $W$, become $J$, $K$ and $W$, respectively. Because of this similarity, the "Teiresias names" will be used for Splash parameters.

Interestingly, although Splash and Teiresias were both developed at IBM there has been, to the best of the author's knowledge, no direct performance comparison between the two applications and certainly no comparison between the two applications on microarray data.

Splash begins pattern discovery by finding a set of *seed patterns*. Conceptually, seed patterns are very similar to elementary patterns in Teiresias and are generated in the same way (see Section 2.5.1). However, Splash goes about extending the seed patterns and checking for maximality in a very different way. Systematically going through each

pattern π in the seed set, Splash checks π for left- and composition-maximality (*lc-maximality*) [Califano, '99]. If π is left maximal, it cannot be extended any further to the left within the limitations set by *L* and *W*. This is checked by "brute force" by referencing the original data set and seeing if any common characters can be appended to the left of the current pattern without reducing π's support. Composition maximality (*c-maximal*) is ensured by checking whether a '.' character in π can be derefernced to a literal. If a pattern is not maximal it is removed from further consideration. In Figure 2.13, it can be seen from Stream 1 and Stream 2 that P1 is not left-maximal because it can be extended to the left by prepending an 'A' character to the pattern without loss of support, nor is P1 c-maximal because the '.' character could be dereferenced to a 'B' character in Stream 1 and Stream 2, resulting in no loss of support.

```
Stream 1: ZABBBCD
Stream 2: ABBBCDH
P1: BB.C {(1,1),(2,1)}
```

**Figure 2.13. Pattern P1 shared between Streams 1 & 2 is neither left- nor composition-maximal.**

After ensuring the maximality of π, Splash attempts to extend π to the right (within the limitations of *L* and *W*) by referencing the original data set in a fashion that is very similar to checking for left-maximality. If π cannot be extended to the right without losing support, π is maximal and is reported. In Figure 2.13, as can be seen from Streams 1 and 2, the pattern "ABBBC" {(1, 1), (2, 1)} can be extend to the right by the addition of the character 'D' with no loss of support. For every extension possible for π (with possible loss of support) a new pattern is formed and added to the seed set, where the patterns will again be subject to maximality checking and extension. Splash continues in this fashion until the seed set is exhausted and all maximal patterns have been found. Califano claims [Califano, '99] that the execution time of Splash is slightly sublinear in the number of patterns discovered. For a more detailed explanation of Splash see the paper by Califano ['99].

When Splash was first introduced it was an improvement over Teiresias because Splash could form *similar* patterns. Similar patterns relax the requirement of exact matching of characters and instead allow matches between sets of characters. Sets of characters, a

subset of the total alphabet, which are considered equivalent are called *equivalence classes*. When pattern discovery takes place using equivalence classes, patterns are allowed to form between characters that are not identical. The output of Splash is slightly changed in this situation; instead of allowing a single character at any position, it is possible to have two or more. In cases where two or more characters (from an equivalence class) form the pattern at a given position, square brackets are used to contain the list of characters found in that position. For the example in Figure 2.14, define three equivalence classes: EQ= {{A, B}, {B, C}, {C, D}}. The resulting aligned pattern is shown below the two sample input streams.

```
Stream 1: ATBBC
Stream 2: ABCAD
Pattern:  A.[BC][AB][CD]
```

**Figure 2.14. An aligned, similar pattern is shared between Streams 1 & 2.**

The use of equivalence classes in microarray data analysis has obvious benefits when considering mapping noisy (or more accurately, "stochastically variable" [Baldi, '02]) floating-point numbers to discreet values (as is needed for pattern discovery). For any binning scheme, 2 arbitrarily close real numbers may be mapped to two different characters even though these two numbers may be considered "close enough" to be the same and justify having a pattern formed between the streams. Allowing characters that represent arbitrarily close numbers to be part of an equivalence class helps to alleviate this binning-precision problem. By allowing two bins that contain numbers that are numerically close to each other to be in the same equivalence class, we allow patterns to form between these close numbers. In Figure 2.15, the following ranges apply to the characters A, B and C respectively: [0.1. 0.2), [0.2, 0.3), [0.3, 0.4). This allows the mapping from Streams 1 and 2 to 1' and 2', respectively. Pattern 1 is the result of pattern discovery without equivalence classes. Note that the values 0.19 and 0.20 (which we will consider close) are mapped to different characters and no pattern forms between the two characters. With the equivalence class EQ= {{A, B}, {B, C}} the pattern AA[AB] appears.

```
1: 0.12 0.15 0.19 0.25
2: 0.11 0.13 0.20 0.35
1′:   A    A    A    B
2′:   A    A    B    C
Pattern 1:  AA
Pattern 1′: AA[AB]
```

**Figure 2.15. Data streams 1 & 2 are binned into 1' and 2' respectively. Pattern 1 forms without equivalence classes, Pattern 1' forms when adjacent bin equivalence classes are used.**

Although the authors make no direct statements as to the performance of Splash they do show that it is significantly faster than Pratt [Califano, '99], an earlier pattern discovery application. However, the time-complexity of Splash seems to be super-linear in the number of patterns discovered [Califano, '99]. For given values of *L*, *W* and *K*, Splash guarantees to find all maximal patterns that fit the density requirement of *L* and *W* for support *K*.

Splash is only accessible through a command line interface and is implemented in the C programming language. It is available for the Sun, AIX and Windows environments and as pre-compiled binaries only.

The other benefit of Splash is that it was implemented early on to work on symmetric multiprocessors (SMP) and cluster environments thanks, in part, to its "embarrassingly" parallel algorithm [Califano, '99]. Although no paper has been written pertaining to the implementation of Splash to run in parallel, it seems possible that each seed pattern could be checked individually for *lc-maximality*. This would allow Splash to be ran in parallel.

### 2.5.3 Genes@Work

Genes@Work is a reimplementation of Splash designed specifically for the analysis of microarray experiments. It is based on the work of Califano et al. ['00]. Therefore, any statements or conclusions pertaining to Splash also bear some relevance to Genes@Work.

Genes@Work takes a three step approach to microarray analysis. Starting with raw microarray data Genes@Work manipulates it, finds patterns in the data (using the same algorithm as Splash), and evaluates those patterns for statistical significance (see below). Finally, Genes@Work groups chips together based on discovered patterns using a greedy covering-set algorithm.

The manner in which Genes@Work works is very rigid. Unlike Splash and Teiresias, which are general pattern finding applications, Genes@Work is a task specific program. Because it closely ties together its data manipulation and pattern finding algorithms, it is very hard to separate the two. For example, it is hard to get Genes@Work to use a different data manipulation technique with its existing pattern finding algorithm and *vice-versa*.

Genes@Work differs from Splash and Teiresias in another, very significant way. Genes@Work not only performs pattern discovery it also implements a s*upervised learning* algorithm in order to perform *classification*. Given a set of data where each element is labeled as being from one of two classes, supervised learning is the act of formulating some scheme that can separate the elements into those classes [Zurada, '92, p94]. This knowledge can then be used to classify future, unlabeled data elements.

Although it is possible to use Genes@Work to only discover patterns (and not to use those patterns to classify the microarray data), the author's desire to make Genes@Work a classification tool influences many aspects of the software. For example, although machine learning techniques could be used to classify both genes and chips, in microarray data it is more common to only classify the arrays. The most common experiment of this type is to classify tissue samples into groups. With this in mind, Genes@Work has been developed to only find patterns between arrays, and not between genes. As a consequence of this design choice Genes@Work only searches for aligned patterns because unaligned patterns are meaningless when they occur between microarrays (see Section 2.3.2.1).

Although classification is often the end-goal of many types of microarray analysis it is not the focus of this work and thus the classification portion of Genes@Work will be explored only in a limited manner. However, because of the classification focus of Genes@Work, it is impossible to remove the requirement that Genes@Work must have a data set that consists of two classes (training data, and test data). This is because training data is central to Gene@Work's binning scheme. The fashion in which Genes@Work performs pattern discovery is detailed below.

Unlike Splash and Teiresias, Genes@Work does not require data values to be binned into discrete characters. However, it does manipulate data values to bring some numerically closer together and separate others, and this can be seen as a type of binning. This binning procedure is implemented using a training data set and an experimental data set. The training data set is arbitrarily picked from one of the two classes of data. The training set is used to build a probability density function $P$ for each gene $u$. This is done by adding together a series of Gaussian distributions which are centered at each expression value in the training set for $u$. An example of $P_u$ is given in Figure 2.16. In this figure the x-axis represents levels of expression and the y-axis represents the probability of a gene expressing at that level. The expression levels of a single gene, $u$, sampled three times, $u_1$, $u_2$ and $u_3$ are shown. Note that many more data points would be needed to compose $P_u$ (not shown).

The probability density function $P_u$, is then transformed so that the distance between two points $p_1$ and $p_2$ in the new scale is equal to the area under the curve $P_u$ between $p_1$ and $p_2$. This new probability density function $P'_u$ is also scaled to have domain [0, 1]. The transformation has the effect of causing $P'_u$ to be uniformly distributed. An example of the $P'_u$ is given in Figure 2.17. In this figure, the data from Figure 2.16 is used. Notice that $u_1$, $u_2$ and $u_3$ have been pushed further apart on the x-axis.

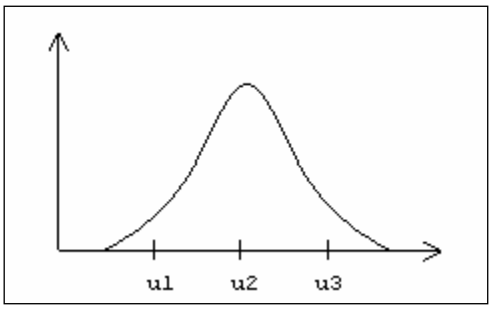


**Figure 2.16. A probability density distribution *P* for some gene *u*, with the amount of expression on the X-axis and its probability on the Y-axis. Three particular expression amounts are shown, u1, u2 and u3.**

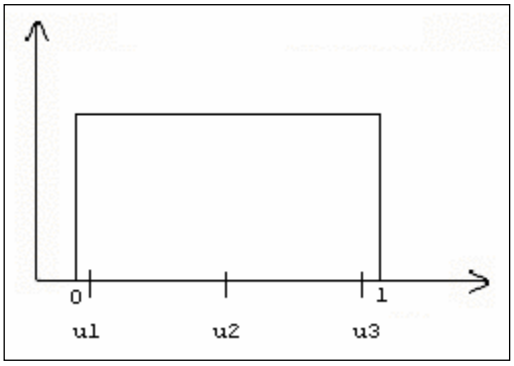


**Figure 2.17. The same probability density distribution pictured above after transformation by Genes@Work.**

The most important aspect of this section is that the transformation function, which was formed using the training data (explained above), is then applied to the test data (the other data class). This transformation which caused the training data to become uniformly distributed, will not cause the test data to be uniformly distributed, unless both data sets come from the same distribution. Instead, some data points will become closer, and others will be pushed further apart, numerically. For example, for each gene $u_{1\ldots n}$, the experimental data ($x_{u1\ldots n}$)can be visualized as points on a number line (for the rest of this example, the subscript *u* will be omitted for clarity). The effect of applying this transformation is as follows: Data points in *x* with values that correspond to a low probability of occurrence in the training data are pushed closer together. Data points in *x* that occur with values that correspond to a high probability of occurrence in the training

data will be pushed further apart. For instance, suppose $x_q < u_1$ and $x_w < ut_q$ and define $d_{qw} = |x_q - x_w|$. Then after transformation $x_q'$ and $x_w'$ are closer together, that is $d_{qw}' < d_{qw}$. Likewise, if we redefine $x_q > u_3$ and $x_w > x_q$ then $d_{qw}' < d_{qw}$. In the other case, where $ut_e \sim = ut_r \sim = u_2$, then $d_{er}' > d_{er}$. That is, $x_e'$ and $x_r$' are now further apart.

Pattern discovery is then executed on the transformed test data. Although the transformed data has been binned, in some sense, it still consists of floating-point data points. That is, for two hypothetical data points 1.0 and 1.2, which become 1.09 and 1.11 after transformation, respectively, they are still not the exact same value and therefore no pattern could form between them. To overcome this, in addition to the standard parameters for Splash, a new parameter, δ, is added to Genes@Work. All values within δ units of one another are considered "the same" and a pattern is formed between them. The δ parameter can be seen as achieving something similar to equivalence classes by allowing close numbers to be considered the same.

The Genes@Work data-transformation process also forms the foundation of its ability to give a probability value to the patterns it discovers. After transformation $P_u$ is uniform (Figure 2.17) and a null-hypothesis is put forward that the genes in the test data, after transformation, are therefore also uniformly distributed. With this null-hypothesis and a δ value one can calculate the probability of finding a pattern composed of $X$ number of genes, with $Y$ amount of support.

The direct integration of this non-linear, gene-specific transformation metric into the pattern discovery process is the major difference between Splash and Genes@Work.

There is no time-complexity analysis available for Genes@Work. However, because the pattern discovery algorithm is based on Splash it seems reasonable to suspect that they are of a similar order of complexity with respect to the number of patterns found in a data set. That said, however, for any particular data set we might expect Genes@Work to be significantly faster than Splash because Genes@Work only searches for aligned patterns.

Although there are potentially an exponential number of aligned patterns in a data set, they represent only a small fraction of the total number of patterns present.

Genes@Work incorporates a number of features to ease microarray analysis. The whole package is accessed through a graphical user interface. Also, Genes@Work allows the user to perform a hierarchical clustering and support vector machine classification on the input data as well as pattern analysis.

Genes@Work is implemented in the Java programming language and is only available in compiled machine code (.class) files. Implementation in Java gives the application good portability to many operating systems. However, there are known memory and speed issues with Java which may yield lower performance of Genes@Work compared to Splash.

### 2.5.4 Comparison

Teiresias and Splash (the basis of Genes@Work) implement very similar algorithms. Both are designed to find generalized patterns in strings and both go about it in a similar fashion. Immediately one is forced to ask: why does IBM support two applications that do essentially the same thing? Is one algorithm better than the other? To the author's best knowledge no comparison of Splash and Teiresias has ever been performed.

Although Splash originally had more utility than Teiresias, since its initial release Teiresias has been modified to incorporate two features of Splash. As stated above, Teiresias now runs in both SMP and cluster environments. Also, Teiresias now allows the user to define equivalence classes. However, an important distinction between Teiresias and Splash is that Splash only allows a maximum alphabet size of 22 characters. Teiresias, on the other hand, allows a maximum alphabet size of 181 characters (all the printable ASCII characters). In addition, a special version of Teiresias allows nearly the entire range of 32-bit integers ($|\Sigma| \sim 4$ billion) to be used as an alphabet. Having an alphabet of just 22 characters can be limiting to pattern discovery if many bins

are used for the stratification of the data and if foreign characters are inserted to break up unaligned patterns (see Section 3.2.2 of Data and Methodology).

When contrasting Genes@Work with Teiresias and Splash it is evident that Genes@Work is a more straightforward program to use for microarray analysis. With integrated pattern discovery and clustering as well as a graphical user interface, Genes@Work has a much shallower learning curve than either Teiresias or Splash. However, these extra features introduce a certain amount of "rigidity" to the analysis process. Effectively, you can only do the exact type of analysis that the authors of the program intended; it is very difficult to "trick" the program into doing something differently. This stands in stark contrast to the relatively open-endedness of Splash and Teiresias, which indeed, were not even originally intended to discover patterns in microarray data.

### 2.5.5 Tupleware

In addition to the 3 other pattern discovery applications mentioned, another pattern-finding program called Tupleware, from Bristol-Myers Squibb [Rogers, '02] exists. Unfortunately, at the present time, only this poster outlining results from the use of Tupleware on serine proteases has been presented. Further, we have been unable to get in touch with the authors in order to obtain a copy of Tupleware or a paper describing how it works. However, it is known that Tupleware builds-up its patterns using a tree structure which its authors believe is more efficient than the lists used by Teiresias and Splash.

# 3 Data & Methodology

Generally, this methodology seeks to further the understanding of the application of pattern discovery to gene microarray analysis. Specifically, it concentrates on two steps in the analysis process, the application/algorithm used to discover patterns in the data and the binning (or preprocessing) of the data for pattern analysis. Three similar but distinct pattern discovery applications will be compared based on ease-of-use and efficiency when used in the microarray analysis domain and four binning methods will be evaluated based on which yields more biologically significant patterns.

This research, and the organization of this chapter, will follow a data-centered (as opposed to an application-centered) methodology. This is done for two reasons. Firstly, biologists do not select an analysis package and then choose what data to collect based on the capabilities of the package. Instead, they have a field of interest and target problem, and need to find a package that can deal with the data that they produce. This target problem can usually be distilled to one of two classes of problems, either the need to relate genes to one another (patterns between rows of data) or the need to relate tissues, or conditions to one another (patterns between columns of the matrix) and this influences what analysis methods as well as binning techniques are applicable. Secondly, the type of data and binning method used dictates what pattern discovery applications can be employed; for example, the two packages Splash and Genes@Work are not applicable to certain types of problems because long data streams tax the limited alphabet of Splash and Genes@Work cannot find unaligned patterns.

Figure 3.1 gives a general overview of the methodology. First, the data is filtered in some way to reduce the number of genes that it contains. In this research, two filtering methods are used: variational filters, which select data streams based on their statistical variance, and *a priori* knowledge, where an expert in the field can pre-select genes known to be important to the problem being answered. Microarray data is filtered because many of the thousands of genes that are monitored by a microarray chip are not interesting for a particular type of experiment. For example, genes that do not change

expression significantly over a course of conditions may not be considered interesting. Also, most microarray experiments contain far too many genes to be effectively analyzed using pattern discovery.

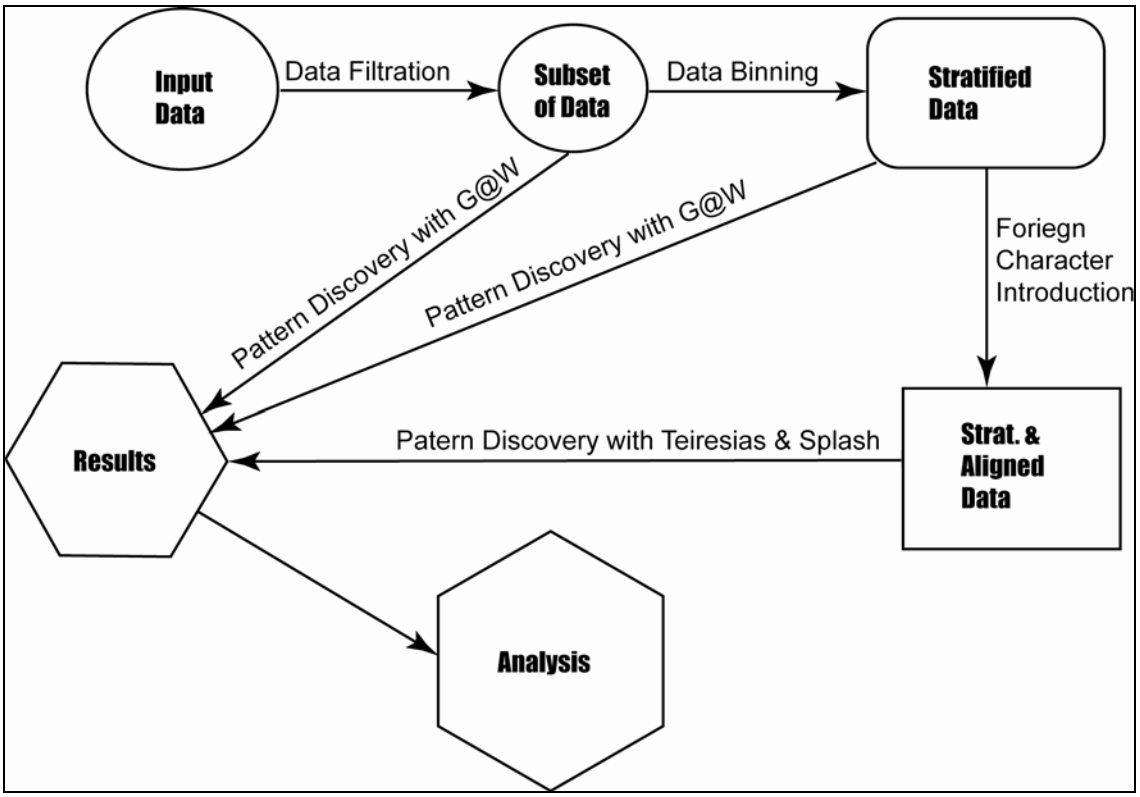


**Figure 3.1. A flow chart of the methodology to be employed.**

Second, the data is manipulated by binning and foreign character introduction. Data binning is needed in order to carry out pattern discovery and in this work four different methods are investigated (although not all ways are equally applicable to all data sets). Foreign characters are introduced into the data streams in order to break up unaligned patterns. This is done because in some cases only aligned patterns "make sense" and breaking up unaligned patterns decreases execution time.

Third, pattern discovery is performed on the data. As stated above, not all applications (Genes@Work, Teiresias, Splash) can be applied to all kinds of data.

Lastly, the observations from the pattern discovery are analyzed. The first step in this analysis is as simple as counting the number of patterns found, or some subset of the total patterns to yield rudimentary results. In many cases these results can be extended to form more interesting conclusions using a more complex form of analysis. Specifically, when expert knowledge can be used to confirm the discovered patterns, the more complex forms of analysis will take place. When no such knowledge exists, simpler methods such as counting the patterns, basic clustering (including visual inspection), or cross validation with other methods will be used. Results from this analysis can be used to validate or refute the various binning methods and determine which applications are more efficient and easier to use.

In the following subsections the two types of data to be used in this research will be explained. This will be followed by an in-depth discussion of the various steps in the methodology as illustrated in Figure 3.1 and the data which needs to be collected at each step. Lastly, this section will discuss other issues to be considered for this research and the materials needed to conduct the research.

## *3.1 Data*

Two sources of data will be used to assess the different binning methods and conduct the comparisons between Genes@Work, Teiresias and Splash: a set of yeast cell-cycle data and a set of related human Diffuse Large B-cell Lymphoma (DLBCL) cancer samples. In both cases the data is the $\log_2$ ratio of gene expression between an experimental state and a base-state. Each of these data sets highlights one of two classes of questions a researcher might ask. Cell-cycle data is used to relate genes to one another (which corresponds to finding patterns between rows of the data). However, because this data is collected over time, specialized procedures can be used with this data, specifically, '+/-/0' binning. The Lymphoma data set is representative of any data collected to relate tissues (not genes) to one another through gene expression (finding patterns between the columns of the matrix).

These two data sets represent a reasonably diverse collection of microarray experiments. Specifically, the data includes both short and long streams as well as different target tissues and organisms, any of which may well influence overall performance of the various applications. Further, these data allow the use of four different binning schemes: the '+/-/0' method, training-data, K-means and the pre-set value method. Lastly, these data not only tie into international interests and knowledge, but also into local work and expertise.

### 3.1.1 Yeast

In general, the question "What does this gene do?" is largely unanswered for a vast number of genes in a huge number of organisms. Gene microarray analysis - specifically, finding patterns between rows of the data matrix - serves as a method to answer this question by relating genes of unknown purpose to genes of known function.

The yeast cell-cycle (ycc) refers to the process of yeast cell growth and reproduction. Although the ycc is well-studied, it is not clear whether all cell-cycle regulated genes have been discovered, nor is it known the exact function of every gene identified as cell-cycle regulated. Clearly these two questions are not specific to just yeast but can be asked of virtually every organism.

The ycc is an ideal subject for analysis because it is a well-studied process and yeast, specifically *Saccharomyces cerevisiae* (hence forth just "yeast"), has a well understood and well-studied genome. Large external data repositories such as GO [GO, '03] and CYGD [CYGD, '04] provide functional information and other annotation data for many of the genes in yeast. These data allow the raw relationships formed from microarray analysis ("information") to be viewed in a biologically significant light ("knowledge"). Further, a well respected ycc data set exists at the Stanford microarray database, the work for which was conducted by the Spellman lab [Spellman, '98]. This data set consists of 6178 genes over 77 conditions. 73 conditions represent 3 separate cell-cycles. The remaining 4 data points are non-cell cycle conditions that show the effect of galactose on

a specific yeast strain. Each cell-cycle uses a different method of synchronizing the yeast cells so that they are all in the same phase of growth.

### 3.1.2 Lymphoma

This data set is representative of data collected to answer questions relating the columns of the data matrix to one another. That is, this data is used to answer the question "which of these tissues is like the others" based on gene expression. In this case, we wish to form multiple clusters (also known as classes or groups) of related but different B-cell lymphomas.

Diffuse Large B-cell Lymphoma (DLBCL) refers to a diagnostic subtype of malignant lymphoma that are similar with respect to their light microscopic appearance and the expression of a limited number of protein markers (for more information see Jaffe ['01]). However, 60% of those diagnosed with diffuse B-cell lymphoma do not respond well to chemotherapy, while the remaining 40% do [Alizadeh,'00]. It has become clear that what is currently diagnosed as diffuse B-cell lymphoma is, in fact, 2 or more separate diseases with similar clinical presentations. A number of microarray experiments have been performed on this type of data in the attempt to separate the multiple kinds of cancer based on gene expression levels [Rosenwald, '02; Wright, '03]. In addition, other methods of identifying markers that can separate subclasses of DLBCL (or predict outcome) have been attempted [Akasaka, '03; Barrans, '04; Hans, '04; Shen '04]. For an overview of discovered predictive markers see Gascoyne ['04]. However, this work will concentrate on microarray analysis and because we are trying to relate tissues to one another (as opposed to genes), patterns will be formed between columns of the data matrix (as opposed to rows).

The lymphoma data set consists of 17,856 genes across 45 different samples of DBCL tissues, including patient samples and cell-lines (raw data and gene lists may be obtained on-line [LLMPP, '02]). In addition, there are 2 samples of normal Germinal Centre (GC) B-cells. These latter two tissue samples have been included because they cluster strongly – in another, larger data set that contains many different tissue samples – with a subclass

of DLBCL. This subclass is then referred to as GC B-like DLBCL and the other is called Activated B-cell like DLBCL.

## 3.2 Methodology

The goals of this research are two-fold. The first goal is to compare four ('+/-/0', K-means, training-data and pre-set value) data binning (stratification) methods to determine whether one produces more biologically significant patterns than the other. As a sub goal of this an approximation of the Genes@Work training-set method will be written and compared to the results of the proprietary Genes@Work application. The second goal is to compare the efficiency, usefulness and ease-of-use of Teiresias, Splash, and Genes@Work.

The efficiency of these applications will be based on both the amount of memory and time (a.k.a. elapsed or run time) used for pattern discovery. When dealing with data sets that produce tens of millions of patterns these two measures are closely related; when memory runs out, the application will either finish very slowly (due to swapping) or, if process space runs out, will simply fail altogether.

The evaluation of ease-of-use and usefulness, however, is largely qualitative and subjective. Ease-of-use will be dependent upon how many parameters there are to set, and how easy it is to determine optimal settings for those parameters. It will also depend on how much data (or other inputs) must be collected (e.g. training data) and/or created (e.g. equivalence classes) in order for the application to function, and how easy it is to create that data. Usefulness will depend upon the functions provided by each application. If one application can do more types of analysis than another, it will be considered more useful. Unfortunately, usefulness and ease-of-use are often inversely related to one another. Generally, the more things an application is capable of, the more parameters there are to set or the more input there is to collect/create.

Not every binning method, nor every pattern finding algorithm, is applicable to every data set. For instance, although the yeast cell cycle data (Section 3.2.5) has only three

binning methods that is applicable to it ('+/-/0', K-means and pre-set value), all 3 applications can be used to find patterns in this data set. This makes the yeast cell cycle data a good data set for application comparison. The lymphoma data set (3.2.6) can be successfully binned by the K-means, pre-set value and training-data stratification methods, but only Teiresias and Genes@Work can successfully discover patterns in the data. In Table 3.1, a summary of which binning techniques can be applied to which input data is presented. The applications that work effectively with the resulting binned data are listed in each cell.

The methodology, as can be seen in Figure 3.1, consists of a number of other steps before pattern discovery can be conducted: selecting a subset of the raw data to conduct analysis upon, binning the data and manipulating the data so that only aligned patterns are found. After pattern discovery takes place, analysis must be conducted on the results to help validate the binning technique used. These steps are discussed in further detail in the following sections.

**Table 3.1. Data, binning techniques and applicable programs.**

| Data Type | '+/ -/ 0' | Training-data | K-means | Pre-set value |
|---|---|---|---|---|
| YCC | Teiresias, Splash, Genes@Work | No | Teiresias | Teiresias |
| Lymphoma | No | Teiresias, Genes@Work | Teiresias | Teiresias |

### 3.2.1 Data Binning and Filtration

Data filtration is an important step in microarray analysis. Most microarray analysis takes place on a subset of all of the genes measured. This is because many genes do not significantly change expression over the various conditions tested. These genes may be "housekeeping genes" (genes that are always expressed at roughly the same amount over the life of a cell) or they may simply be genes that are not part of the phenotype being tested (e.g. non-cell-cycle related genes) and thus do not change significantly from

microarray to microarray. Often the subset of genes that are selected is based on a priori knowledge about the problem area. When no such knowledge is available, genes can be selected based on their variation over the conditions tested [Califano, '00]. That is, genes whose statistical variance exceeds some arbitrarily selected threshold are chosen for analysis. The reasoning behind this approach, termed a variational filter, is that genes that vary a great deal over the various conditions tested are more likely to be "interesting", that is, they are more likely to be genes that contribute to the phenotype in question. It is important to realize that this is not always the case. Genes with low variability may still be important to the researcher, and will not be considered when using this technique.

For every data set used in the research, when a subset of the full data is needed, the genes used will be selected via the above described variational filter. The threshold will vary for each data set based on how quickly patterns can be discovered. In general, all pattern discovery should take between 1 and 24 hours of run time for patterns with some reasonable support amount (K=2-6) when using Teiresias. Other applications, when using this same data set may take a longer or shorter amount of time to finish.

Before pattern discovery can take place on the filtered data, the data must also be binned: floating-point, continuous data must be mapped to discrete characters in order to allow pattern discovery.

In this research four data binning techniques will be used; K-means, '+/-/0', the training-data technique (and its approximation), and the pre-set value technique. Each technique has its own difficulties, but each may also have its own benefits in detecting different kinds of biological relationships (See Sections 2.4 – 2.4.4).

The primary problem associated with all stratification (binning) techniques is selecting the appropriate number of bins to use to discretize the data. The formation of too many bins results in no patterns being found (too much granularity); too few bins may result in many spurious patterns being found. Since choosing the correct number of bins (or other

input parameters) may be difficult, this has a strong effect on the usefulness of a binning method. Each of the four stratification techniques used here also has this problem. The K-means binning method requires the researcher to select number of bins to use, K. Similarly, in the case of Genes@Work (the training-data method) it is the researcher who must come up with an appropriate value for δ, which is related to the number of bins to use. However, the '+/-/0' and pre-set value methods try to circumvent this problem by simplifying gene expression as transitioning between any of a small number of states (usually 3 or 5)[6]. However, there is no reason to believe that all genes conform to this simplified view of gene expression and these latter two techniques may poorly model more complicated forms of expression.

When attempting to bin values using the K-means clustering method, *K* different bins result. However, the selection of *K* is generally based on *a priori* knowledge. As no such information is available for most microarray work, the Xmeans clustering program will be used instead [Pelleg, '03 a]. Xmeans is simply a program that executes multiple runs K-means clustering on a single data set using different values of *K*. Xmeans determines which value of *K* was "ideal" based on a Bayesian Information Criterion (see Section 2.5.4). Initial work using Xmeans with the data described in Section 3.1.2, shows that Xmeans will pick a *K* value between 3 and 16 for most genes. These K-values seem reasonable (neither too high nor too low). Further, the value of *K* selected is very robust with respect to varying input parameters. For instance, when allowing the possible values of *K* to be from 2 to 20 (where $20 \sim |V|/2$, see Section 2.5.4), Xmeans often picks the same value for *K* as when allowing *K* to range from 2 to 50.

Another binning method, termed the "+/-/0 technique", is most applicable to time series data [Rigoutsos, '00]. The choice of which bin to place the current data point is dependent on whether the slope of the line between this time point and the next is increasing, decreasing, or staying the same. That is '+' for a positive slope, '-' for a negative slope and '0' for a slope of zero, or more generally, when the absolute value of

[6] The reasoning behind these numbers is the belief that genes can only increase, decrease, or have no change in expression. Occasionally, "very large increase" and "very large decrease" are added to the allowable states.

the slope is below some chosen threshold ("close enough" to zero). It is the value of this threshold, however, that may be difficult to determine.

When using the Genes@Work approximation (training-data method) the number of bins to use is inversely proportional to δ. Califano et al. ['00] advocate δ values between 0.05 and 0.15 which corresponds to 40 to 13 bins, respectively, under the approximation in Section 3.2.4. 40 bins is far past the capabilities of the publicly available versions of Splash (which only allows 22 characters, and therefore a maximum of 22 bins), and therefore a smaller number of bins will be used. However, it is also recommended [Califano, '00] that δ values be set such that 10 to 20 statistically significant patterns are discovered. Therefore, the number of bins to use will be adjusted until the number of statistically significant patterns, as determined by Genes@Work, is between or close to 10 to 20.

Determining the delimiters for the pre-set value technique is done arbitrarily. However, using the delimiters {-1.00,-0.25,+0.25,+1.00} leads to many values in gene expression data to be considered unchanged, and only a few to be considered greatly increased or greatly decreased and this seems reasonable.

### 3.2.2 Aligned and Unaligned Patterns

Although aligned and unaligned patterns occur in the data sets, only aligned patterns will be considered. Aligned patterns, patterns that occur at the same offset, are meaningful for both the yeast cell-cycle and lymphoma data sets. Unaligned patterns, which can occur with any combination of offsets, can have meaning for time series data (such as yeast cell cycle data) will not be sought and are beyond the scope of this work.

Teiresias and Splash are general pattern finding applications and can find both aligned and unaligned patterns. However, by introducing characters common to all streams at short intervals, one can "break up" unaligned patterns [Rigoutsos, '02]. By doing this and focusing solely on aligned patterns, one can reduce execution times greatly because unaligned patterns generally form the majority of all patterns in a data set. In Figure 3.2, the "raw" data is present on the left while the transformed "break" data is present on the

right. Notice in the left-hand column a single unaligned pattern (in bold) exists in streams 1, 2 and 3. The occurrence of this pattern in streams 1 and 2 is left intact after the addition of foreign characters, whereas the pattern is broken up in stream 3. If the interval at which the foreign characters is introduced is *Z*, then all aligned patterns can be found with *L*=2, *W*=*Z*+2, *K*=2 (see Section 2.5.1 for explanation of these parameters). Ideally, a different foreign character is used at each insertion position in the streams. When this is not possible it is important that the characters not appear repeatedly in the same order.

| | |
|---|---|
| Stream 1: **abcde** uiw | 1 **abcd** 2 **e** uiw4 |
| Stream 2: **abcde** tqk | 1 **abcd** 2 **e** tqk4 |
| Stream 3: yy **abcdeq** | 1yyab2cdeq4 |

**Figure 3.2. Three streams before and after foreign character insertion to demonstrate the breaking-up of unaligned patterns.**

The spacing at which to introduce the foreign characters is a straightforward matter to determine. The base case, where *Z*=1, will break up the most patterns and have the greatest effect on execution time . However, many streams are simply too long or an alphabet too small to introduce a foreign character between every other data character. A compromise of *Z*=4 will be used for the yeast data set, but *Z*=1 will be used for the lymphoma data set.

### 3.2.3 Training Data

This section briefly outlines a number of problems encountered when analyzing gene expression microarray data and how the use of training data, as is done in Genes@Work, can help alleviate them. It concludes with a short discourse on several problems encountered when using training data for binning microarray data with a primary focus on the selection of the training set.

Regarding gene microarray experiments, it is intuitively understood that the researcher is interested in genes that are responsive to changes in the environmental condition (herein termed *excited* genes). That is, genes whose expression remains "unchanged" across the

varying environmental conditions are likely not important to the researcher because they do not contribute to the phenotype of the cell in question. However, what is truly meant by "unchanged" is not necessarily that the expression of the gene does not change at all, but rather that if the expression does change, it does not change unexpectedly. A simple standard deviation filter (see Section 3.2.1) , allows genes that change expression to pass through, even if the changes may be expected. Thus the Training-Data technique can be seen as a refinement on this more simple filtering method. What is considered "expected" for a gene is how the gene acts in some predefined, application specific state or set of states which are considered "normal". In Figure 3.3 we see two genes across four conditions. The expression levels of the first gene do not change significantly across the conditions; however, it is unknown whether even this small amount of change is a significant amount of variation for this gene. The second gene has radical fluctuations in its level of expression; however, it is possible that these fluctuations are no different than what normally occurs when this gene is in its base (or normal) state.

```
g1: 1.01 0.93 1.10 1.05
g2: -0.87 1.05 0.24 -0.11
```

**Figure 3.3. Two genes with very different expression profiles under some experimental condition.**

The problem now becomes how can a researcher tell whether a gene is excited or not. A potential solution is to use a data set to train a computer to differentiate between an excited gene and a non-excited gene. A *training set* consists of data from a number of base (normal) states for the cell in question. For every gene *g* in a given experimental data set *E*, *g* also exists in a training data set *T*. $T_g$ is a vector of data points consisting of expression values for gene *g* in its base state. Every gene in *E*, $E_g$, is trained against $T_g$. This means that the training that takes place is gene specific and a consequence of this is that a single value can be treated differently if it occurs for two different genes. More simply, $T_g$ provides the computer an understanding of how gene *g* normally acts and *ergo* whether *g* is acting abnormally in data set *E*. In Figure 3.4 the training data for the two genes in Figure 3.3 is provided. We can see from this data that the slight changes in *g1* are unexpected (because the training data hardly changes at all), and the fluctuations in

*g2* are normal because the gene in its normal state alters its expression as much as it does in the test states.

```
g1: 0.90 0.93 0.91 0.90 0.89
      0.93 0.92 0.95 0.88
g2: -1.02 0.72 1.22 -0.53 0.55
      -0.35 0.10 -0.22 0.88
```

**Figure 3.4. Gene expression values for two genes from the previous figure in their base state.**

This concept is presented by Califano et al. ['00] in a slightly different manner. In Figure 3.5 we see the probability density function for some theoretical gene in its normal state. The x-axis represents the expression level of the gene, and the y-axis represents the probability of expression occurring at that level. Such a probability function can be constructed empirically from *T*. The goal of training is to use this training data, in effect, to indicate to the computer that gene expression as low as *u1* is unexpected, expression in the range around *u2* is common, and expression as high as *u3* is also unexpected.

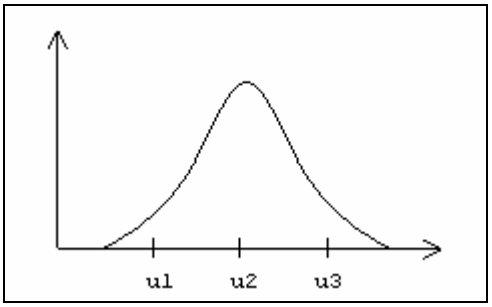


**Figure 3.5. Probability density distribution of a gene's expression values with three expression levels indicated.**

When considering training in the context of pattern discovery, the goal is to stop patterns from forming between unexcited genes and encourage patterns to form between excited genes. The manner in which to do this is to allow a wide range of values to be placed in a single bin when expression in that range is unexpected and to allow only a small range of values to be placed in a single bin when expression in those ranges is expected . To state this in another way, we use the training set to provide fine granularity binning in high

expression probability ranges, and coarse granularity binning in low probability ranges. To illustrate this, a more complex example is provided in Figure 3.6. In this Figure we see the experimental data for a single gene $E_G$ and the training data for the same gene $T_G$. At first, it might be concluded that the values 1.10 and 1.20 in $E_G$ are virtually the same value and should be placed in the same bin, while the values 4.00 and 6.00 are significantly different and should be placed in different bins. However, when the training data is considered we see the opposite is in fact true; the values 1.10 and 1.20 are expected (the training data contains many values in this range) and thus should be mapped to different bins and the values 4.00 and 6.00 are unexpected and thus are mapped to the same bin. This data set is said to provide a fine granularity in the range of ~1.00 – ~1.30, because so many data points fall into that range in $T_G$, and a coarse granularity everywhere else. With regards to $E_G$, $T_G$ implies that even though 1.10 and 1.20 are numerically close together, they are significantly different and that although 4.00 and 6.00 are numerically distant, in this context they should be considered that same. Vernacularly, 4.00 and 6.00 can be labeled "abnormally high", and instead of overly selective they are mapped to the same bin. The opposite also holds: if the values -3.00 and -4.00 appeared in $E_G$, then a similar mapping would take place. That is, -3.00 and -4.00 should be considered "abnormally low" and mapped to the same bin.

```
E_G:  1.10  1.20  4.00  6.00
T_G: 1.01 1.06 1.12 1.16 1.16 1.20 1.24
```

**Figure 3.6. Experimental and training gene expression data**

To fully appreciate the effects of $T_G$ in Figure 3.6 on $E_G$, suppose that each data point in $E_G$ is from a different tissue. By mapping 4.00 and 6.00 to the same bin, patterns are encouraged to form between these tissues due to the abnormally high expression values for this gene. At the same time we discourage patterns from forming between the first two tissues because these genes are not acting unexpectedly. The benefit to the latter effect is that if most genes act normally in a given experiment, then by discouraging patterns from forming between these genes the total number of patterns found is reduced and thus the time required for pattern discovery lessened.

Despite the benefits of using training data to stratify the experimental data there are also a number of pit-falls of which to be wary. Firstly, the number of data points in the training set should exceed the number of points in the experimental set. Califano ['00] uses at least twice as many training samples as experimental. The second concern is that the type of data used for training, what the researcher considers a base state, will influence what patterns are found. Using a base state that is too similar to the experimental state will cause most patterns to be destroyed, potentially destroying significant patterns, while using a base state that is too dissimilar to the experimental state will cause many uninteresting patterns to be formed. Califano ['00] uses different subtypes of cancers from the NCI-60 cancer data set. Although all the data samples are related (cancers) subgroups can be formed based on a number of factors (cancer type, mutation type, reactivity to drugs). The goal, generally, is to use a base state that has similar expression for genes that are not involved in the phenotype of the cell (unexcited genes), and for the excited genes, to have differing expression levels to encourage meaningful patterns to form with those genes. However, to do this perfectly, the researcher would have to know *a priori* which genes were involved in the phenotype of the cell in question which, unfortunately, is the very question the researcher is attempting to answer in the first place. Therefore, the best that can be hoped for is the researcher making an educated guess at what constitutes a "good" training set.

Selection of the training data for pattern discovery in gene microarray analysis is a very new problem and is currently left to experts in the area of research. Although study in this topic area – the effects of different training sets on discovered patterns – is very important to the use of this technique, it is beyond the scope of this research. Instead, when a training set is required for this research, the selection of the data will be made with the advice of an expert in the relevant field.

### 3.2.4 The Genes@Work (Training-Data) Approximation

Section 2.5.3 describes Genes@Work's method for binning and discovering patterns in microarray data. Replicating this form of pattern discovery for Teiresias or Splash is not straight-forward. The first step in the Genes@Work application, transforming the input

based on the training data, is mathematically complex and is not as easy to implement as presented [Califano, '00]. However, this is less important than the fact that neither Teiresias nor the publicly available version of Splash can find patterns in continuously valued data (the initial state of microarray expression data). That is, they cannot be made to form a pattern between two numbers $a$ and $b$, where $|a - b| < \delta$.

It is possible to approximate both the data transformation and continuous pattern discovery. It is important to note that this approximation is not perfect and will likely not yield the same results as Genes@Work. The approximation is more strict than Genes@Work. That is, it may result in the failure to find some aligned patterns that are found by Genes@Work; however, it should not result in finding aligned patterns that are not found by Genes@Work.

A way to approximate the above transformation is to divide the training set into $\lceil 2/\delta \rceil$ bins. Each bin is defined by a range of numbers that fall into the bin. Thus the $n^{th}$ bin $B_n$ is defined as all values $V$ that satisfy $D_n < V \leq D_{n+1}$. The numeric delimiters ($D_1, D_2, D_3, \ldots, D_{\delta*2}$) used to form these bins are then applied to the experimental data. Although the number of items in each bin is equal, the values for $D_i$ will not be evenly spaced unless the distribution of $V$ is uniform. This has the effect of inducing a fine granularity in ranges where the training set has many values, and a coarse granularity in ranges where the training set has few values (see Sections 3.2.3 and 1.9). In Figure 3.7, an example input stream is grouped equally into 4 bins ($\delta = 0.5$). The character representing the bin and the delimiters for that bin are also shown. In Figure 3.8, the delimiters from 3.2.4-1 are used to bin the input stream shown. Note that even though the input is uniformly distributed the output highly favors the "D" bin. This is because values over 1 are so rare in the training data that they are considered "unexpected" and thus are "compressed" into one bin in the "test" data.

The final step in the approximation is to execute pattern discovery using a set of equivalence classes $EQ=\{\{B_0,B_1\}, \{B_1,B_2\}, \{B_2,B_3\},\ldots\}$. EQ allows two bins to be effectively considered as one, and because the width of each bin is $\frac{1}{2}\delta$, the maximum

difference between 2 numbers (from the training set) mapped into the $n^{th}$ equivalence class is δ (after transformation). This accomplishes a similar task as forming a pattern where $|a–b| < \delta$.

```
Training Data: -4 -2 -1 -0.5 -0.3 0.0 0.1 0.5 0.75 2 5 8
A: (-infinity, -1]
B: (-1, 0.0]
C: (-0.0, 0.75]
D: (0.75, +infinity)
```

**Figure 3.7. Ranges for training data when sued to create 4 bins.**

```
Test Data:
-2 -1 0 1 2 3 4 5 6 7 8 9
Test Output:
A   A B D D D D D D D D D
```

**Figure 3.8. Test data and resulting characters using the training data and bin ranges from previous figure.**

### 3.2.5 Yeast Cell-cycle

The data for this section is described in Section 3.1.1.

Analysis of the yeast cell-cycle data traverses all the steps present in Figure 3.1. First, the size of the data set will be reduced. It will then be binned using the '+/-/0' (Section 2.4.2) , K-means (Section 2.4.3) and Pre-set value (Section 2.4.4) methods[7]. This data will be used for aligned pattern discovery using Teiresias, Splash and Genes@Work. The goals of this analysis are two fold, to benchmark each of the pattern discovery applications against one-another and to compare the biological significance of each of the binning techniques. Patterns will be found across the genes in this data, that is, the rows of the data matrix. Finally, the discovered patterns will be assessed for biological significance, the two binning methods will be compared based on how many significant patterns are found using each technique, and the efficiency of the 3 applications will be compared.

[7] Neither Genes@Work nor its approximation will be used due to a lack of training data in this data set.

Because 6178 genes is too large a data set on which to practically run pattern discovery, and because many genes will not modulate expression significantly over the course of a cell-cycle, a subset of 612 genes (~10%) over all 77 conditions will be chosen using a variational filter (see Section 3.2.1). The number of genes selected for analysis will depend on the memory requirements for pattern discovery. That is, using the following parameters, $K$=2 and $\varepsilon$=0.2 (see Section 2.5.2), pattern discovery should consume less than 4GB of process space (see Section 3.1.1). This data set will be considered a "moderate number of streams over a moderate number of conditions (77)".

Because of the nature of time-series microarray data, unaligned patterns do make biological sense. However, this work will concentrate on aligned patterns only. Aligned patterns will be discovered using $Z$=4. Because of the large number of genes a density of $L$=4, $W$=9 will be used (see Section 2.5.1). For each input data set, pattern discovery will take place using different values for minimum support. Specifically, $K$=2, $K$=3 and $K$=6 will be used.

Unfortunately, preliminary work on this data for benchmarking the three programs revealed that neither Splash nor Genes@Work could finish execution on 612 genes with the above parameters. Further, it was shown that Splash did not find the same patterns as Teiresias (see Section 4.1.1 for more detail). However, both Splash and Teiresias could find the same patterns when exhaustive pattern discovery parameters were used. To provide Teiresias, Splash and Genes@Work with workable input, 4 data sets were constructed from the yeast cell-cycle data using the first 25, 30, 35 and 40 streams from the 612 streams used in the remainder of this methodology. Each data set was constructed using a break value of $Z$=4 (see Section 3.2.2), which means with $W$=6 and $L$=2, all patterns can be discovered in the data set for a particular support, $K$. For all benchmarking data sets $\varepsilon$ =0.2 was used because, as will be shown in Section 4.1.4, pattern discovery is fastest with this value of $\varepsilon$. For each data set 4 values of $K$ were used: 2, 3, 4, and 5 and their time and space utilization was recorded.

This data will be binned using 3 methods: K-means , '+/-/0' and pre-set value. The K-means binning will be performed by the Xmeans application. Although Xmeans automatically selects the number of bins to use for each gene, this number will be forced to lie in the range of [1… 48][8]. Data from the K-means binning scheme will be analyzed by Teiresias only. This is because K-means binning requires more characters than Splash is capable of representing and it is very difficult to get Genes@Work, with its integrated binning scheme, to form the same bins as K-means.

There are two issues to contend with when using Xmeans to bin gene's expression values, when intending to discover patterns between the genes. First, two genes that are binned with different values for *K* are difficult to compare; If one gene has 20 bins, and another has only 2 bins ("high" and "low") then it is impossible to instruct any of the pattern discovery programs used here to allow the single "low" valued bin in one gene to associate with the X-number of low valued bins in another gene. For example, in Figure 3.9, two streams are shown before and after Xmeans binning. The first stream has two centroids, one for the positive values, and another for the negative values. Thus, in Stream 1', 'A' means up-regulation, and 'B' encodes down-regulation. In Stream 2' there is three centroids. 'A' still encodes up-regulation, but 'B' encodes normal-regulation and 'C' encodes down-regulation. Thus, if a pattern were to form between Stream 1' and Stream 2' using the character 'B', then it would be meaningless because the 'B' means two different things in each stream.

```
Stream 1: 5.5, 5.0, 5.2, -5.0, -5.2, -5.5
Stream 2: 8.0, 7.9, 0.2, 0.1, -2.0, -2.2
Stream 1': A, A, A, B, B, B
Stream 2': A, A, B, B, C, C
```

**Figure 3.9. The effects of different values for *K*, the number of centroids**

The second problem is that with a high value of *K* streams become too finely binned. If two streams are both binned with 20 centroids it is very unlikely they will form any

[8] A value of 48 means that less than 2 genes need to be in each centroid. Also, it was predetermined that no genes are ever mapped to more than 36 centroids. Therefore, this value is effectively unlimited.

patterns, because on average, any particular bin is only represented by 4 (of 77) characters.

Fortunately, both problems can be solved by re-mapping the binning scheme provided by Xmeans to use an equal (or near equal) number of bins for all genes. Under the assumption that all genes are either up, down or normally expressed we re-map the centroids Xmeans finds to 3 bins[9]: 'A', 'M' and 'Z' – up, normal and down. If Xmeans finds 2 centroids, then they are mapped to 'A' and 'Z'. If `K(mod3)=0`, then the centroids are split evenly into 3 bins. If `K(mod3)=1`, then the extra centroid is placed into the "normal" bin. If `K(mod3)=2`, then the extra centroids are placed in the "high" and "low" bins. For example, if Xmeans finds 4 centroids (1, 2, 3, 4) then 1 gets mapped to 'A', 2 and 3 get mapped to 'M', and 4 gets mapped to 'Z'. If Xmeans finds 5 centroids, then 1 and 2 get mapped to 'A', 3 gets mapped to 'M', and 4 and 5 get mapped to 'Z'. In Figure 3.10, we see the data from Figure 3.9, after the bins have been re-mapped to 3 bins. In a simple case such as this, the number of bins for each stream do not change. However, Stream 1'', 'B' has been re-mapped to 'Z', meaning that under those three conditions the gene expresses at its lowest levels. This makes the meaning of each character in-line with the characters in Stream 2''. Thus, any pattern that forms between Stream 1'', and Stream 2'', will have meaning.

```
Stream 1':  A, A, A, B, B, B
Stream 2':  A, A, B, B, C, C
Stream 1'': A, A, A, Z, Z, Z
Stream 2'': A, A, M, M, Z, Z
```

**Figure 3.10. K-means binning data before and after merging into 3 bins**

Note that merging the bins is different from restricting the upper limit of *K*. When the upper limit of K is restricted to some low value, then many genes are mapped to a single

[9] Although 3 bins were chosen 5 bins (similar to the pre-set value technique below), or 2 might also be reasonable choices. However, as the number of bins is increased, fewer patterns may be found, and those patterns will have lower densities, whereas using only 2 bins may cause many patterns to be discovered and cause a large increase in execution time

bin. For instance, with K limited to 5, nearly half of the 612 genes have a single bin, whereas with a limit of 48 (which effectively is unlimited for this data set), only 2 genes are mapped to a single bin.

Initially, this technique may appear to be disposing of the work that Xmeans has done. However, this is not the case. Although some information loss does occur in mapping a large number of bins to 3, Xmeans is still very critical to this approach and results in a very different binning from the other techniques. This technique still relies on Xmeans to determine whether there is any change in gene expression at all. If Xmeans determines that the data can be accounted for by a single centroid, then as before, the gene is not used under this technique. Further, the range for each bin is determined by Xmeans. In Figure 3.10, 'A' is used for data values between 5.0 and 5.5 for Stream 1, and 7.9 to 8.0 for Stream 2. Thus Xmeans gives stream-specific (in this case gene-specific) delimiters for each bin, unlike the pre-set value technique which uses global delimiters for each bin.

The '+/-/0' technique has been successfully used for binning of YCC data by Teiresias [Rigoutsos, '00]. However, Rigoutsos ['00] makes no recommendations for $\varepsilon$, therefore 3 separate data sets will be formed using the following 3 $\varepsilon$ values: 0.1, 0.2, and 0.4.

Finally, the pre-set value technique will be used with the values {-1.00, -0.25,+0.25,+1.00}. All three pattern discovery applications can be used to analyze each of the two data sets: this technique was originally developed using Teiresias, the small number of characters required for this type of binning permits Splash to be easily applied to this kind of data (as Splash only supports a small, 22 character alphabet size) and Genes@Work is very capable at discovering patterns in this type of data. It should be noted that it is slightly more difficult to get Genes@Work to use the '+/-/0' binning technique in place of its built-in binning scheme, however it is possible and the procedure for manipulating Genes@Work into doing '+/-/0' binning is outlined below.

Although Genes@Work was not originally intended for '+/-/0' binning it can be coerced into doing so through a two step process. Normally, in the first step, the floating-point

data is transformed into '+/-/0' input. However, as Genes@Work will only accept numeric input, the data is instead transformed to "a large positive value", '0' and "a large negative value" – for instance +10, 0, and -10 – in place of '+', '0' and '-', respectively. The next step is to ensure that Genes@Work will not form patterns between these different values. That is, no value of delta should be large enough to allow +10 and 0 to be considered "close enough" to form a pattern between them even after normalization with the training set (see Section 2.5.3). This is accomplished by using a small value for delta is used and by contriving a special training set is constructed that clearly separates -10 from 0, +10 from 0 (and *ergo* -10 from +10). Such a data set has a large number of data points that lie between -10 and 0, and 0 and +10. For instance, many data points of -5 and +5 would suffice. Such a training set causes Genes@Work to regard -10 as "very low and unexpected", 0 as "intermediate and unexpected", and +10 as "high and unexpected". By using enough data points in this contrived training set, one can ensure that +10, 0 and -10 are all treated "atomically". The final question to answer is how many contrived data points are needed to separate the three bins. This value should be high enough to ensure that +10, 0 and -10 are all well-separated no matter how many data points are in any of the +10/0/-10 bins. By using the same number of points for +5 and -5 as there are in the original data set (77), we ensure separation even in the worst case by guaranteeing that there are more data points in the "+5" and "-5" bins then there are in any other bin.

To conclude, in order to achieve '+/-/0' binning, Genes@Work is provided with specially contrived training data that consists of many (77) instances of a single "intermediate positive value" (+5) and many (77) instances of a single "intermediate negative value" (-5). From this, Genes@Work will effectively break the data up into 3 bins (a "+10 bin", a "0 bin", and a "-10 bin") . Pattern discovery is then executed on this data with some very small (near zero) value for delta that does not allow patterns to form between the 3 bins.

All pattern finding techniques will be used with the '+/-/0' data so that timing and memory benchmarks can be performed. Once that data is collected however, only Teiresias will be used with the pre-set value and K-means binned data.

To monitor the efficiency of Teiresias, Splash and Genes@Work the running time of each execution will be recorded using the UNIX program "time", which measures the full execution time of a process. "time" will be considered accurate for execution times of 100 seconds or more. For each execution the UNIX program "ps" will be used to poll the memory footprint of the process every minute. This interval will provide many time points for a given execution, and yet will not overly tax the CPU.

After execution, it will be confirmed that the longest patterns in the '+/-/0'-binned data discovered by Teiresias are also discovered by Splash and Genes@Work. In addition, these patterns will be assessed for their biological significance. This will be done by comparing the discovered patterns with known gene relations in the GO [GO, '03] CYG [CYGD, '04] databases. Only clear gene relationships that are well-established will be considered "hits". Other relationships, although not necessarily "misses" will not be considered further as they would require biological expertise to confirm their validity and this is not the focus of these experiments.

In addition to comparing the execution times and memory usage of Splash to Teiresias, the change in execution times with varying support levels (2, 3, 6) will be examined. Also, the change in execution times and the number of "hits" achieved with various $\varepsilon$ values (0.1, 0.2, 0.4) will be examined.

### 3.2.6 Lymphoma

Section 3.1.2 describes the data used for these experiments. The first step in this investigation is to select a subset of genes for analysis. Fortunately, this has already been accomplished by Alizadeh ['00]. This smaller data set will be binned using three techniques, K-means, pre-set value and the Genes@Work training-data method. Pattern discovery will then take place using Genes@Work – which will use its built-in binning scheme – and Teiresias which will use the Genes@Work approximation, pre-set value and the K-means binned data. The patterns sought in this type of analysis are those that occur between the columns of the data matrix. That is, we wish to relate tissues to one

another based on how similarly the genes express in those tissues. Finally, the resulting patterns from each binning technique will be examined for biological significance. Genes@Work and Teiresias will be compared in terms of memory efficiency and execution time. The success of the Genes@Work approximation to accurately approximate the results of Genes@Work will be determined.

Firstly, it must be determined which genes should be used in this analysis. Alizadeh et al. [Alizadeh, '00], used a subset of 380 genes (that represent the germinal centre B-cell signature) of the 17,856 genes present on the microarrays used in this experiment to separate the two classes of disease from one another. Although they later identify more genes that seem to yield a clearer separation of the two cancer types, this work will only consider the 380 genes identified in order to make the results more comparable. This analysis will be considered "a small number (47) of long (150 character) streams".

The combination of long data stream length and the need to add foreign characters for aligned pattern discovery will stress the small alphabet available with Splash and thus Splash will not be used in this analysis. However, Genes@Work only searches for aligned patterns and is well suited for this analysis. Teiresias, because of its large available alphabet will have little problem dealing with the long stream lengths involved in this data set and will also be used to discovery patterns.

After the data is selected it will be stratified in four different ways, using the Genes@Work approximation (for Teiresias) or Genes@Work's built in binning scheme, Xmeans, and pre-set value. The Genes@Work approximation will take place with 10, 12, and 14 bins, which correspond to delta values of 0.20, 0.167, and 0.143, respectively. The data binned using Xmeans will be for use with Teiresias only as it is difficult to get Genes@Work to form these same bins. When Teiresias is used, foreign characters will be introduced into the data at an interval $Z$=1. Thus, $L$=2, and $W$=3 will allow all patterns of support $K$ to be discovered (See section 2.5.1). For both Genes@Work and Teiresias, $K$=2 will be used. Because Genes@Work is not implemented to work on a

cluster, it will be executed on a Solaris machine (see Section 3.3). However, in order to save time Teiresias will be executed on the Beowulf Cluster (see Section 3.3).

It will be examined how well pattern discovery, using Teiresias, can cluster this data using Xmeans and pre-set value as stratification techniques. An important distinction should be made between the use of Xmeans here and the use of it with yeast cell cycle data. Previously, Xmeans was used to stratify the numeric data into characters on a gene-by-gene basis, then pattern discovery was used to find patterns between the genes. Because of this it was important that characters from one gene be comparable to another (i.e. an 'A' in gene 1 should essentially mean the same thing as an 'A' in gene 2) and in order to do this, when Xmeans found more than 3 bins for a particular gene, the bins were "compressed" into just 3. That problem does not exist with this data, however, because pattern discovery takes place between the arrays, not the genes. Therefore, the binning scheme used for one gene is independent of the binning scheme used for any other gene. In the extreme, an entirely different set of characters could be used for each gene and the pattern discovery would still work. Contrariwise, if such a thing were done with the yeast cell-cycle data, no patterns would be discovered at all.

There is another important distinction between this work and that done with the yeast cell-cycle. Because the yeast cell-cycle data set consists of time-series data, it could be expected that the patterns would be dense. That is, once a set of genes begins co-expressing, they are likely to continue co-expressing in the next time instance. However, in this data the order of the genes is random with respect to any patterns they might form. That is, the first and last genes are just as likely to form a pattern as the first and second. Therefore, it is absolutely critical that exhaustive pattern discovery can take place. This means that any aligned pattern discovered consists of every co-expressed gene shared between the supporting tissues. Using $Z$=1 allows the least number of unaligned patterns to form. However, with such a small value of $Z$, Teiresias must reuse all the break-characters approximately 4 times. This will cause a number of unaligned patterns to be discovered, but it was determined this was still faster than using $Z$>1.

Because of the length of the streams, the upper value for the number of centroids will be limited in order to conserve the number of characters needed for data binning. To test the effect of limiting the number of centroids four values will be used: 5, 10, 15 and 20.

The approach used to evaluate the quality of the patterns is a three step process. First, a number of different criteria will be employed to score the patterns. Each scoring scheme is an attempt to balance the number of genes in the pattern (also referred to as the length or density of the pattern) with the support of the pattern, a feature that was shown to be critical in the yeast cell-cycle results. Second, the top 20 patterns[10] for each scoring scheme will then be picked "blindly". That is, without regard to the proportion of a particular subtype that supports the patterns. Third, the quality of these blindly-picked patterns will be evaluated based on how well they agree with the subgroups found by Alizadeh. That is, if the vast majority of the tissues which support the pattern are of either ABC or GCB subtype then the pattern will be defined as a good classifier. This approach will give some insight into the minimum number of genes and minimum amount of support a pattern would need to be considered a good classifier. The underlying hope is that it might prove to be a general rule that could be applied to many kinds of data. After the patterns that are best able to classify the two subgroups have been determined, they will be used to form a list of genes (candidate genes) that seem to be most important in separating the two classes. Afterwards it will be considered whether the classification of some tissues by Alizadeh is incorrect and the implications that has on analysis.

Analysis of the Genes@Work results will be slightly different. Because Genes@Work is primarily a classification tool it must be told to which of the two classes any particular array belongs. It then uses one class as the training data, and searches for patterns in the other class. This means that any pattern discovered by Genes@Work will consist entirely of one class. The way in which Genes@Work will be compared to Teiresias, then, will be in terms of the length and support levels of the most important patterns as determined

[10] Certainly more than 20 patterns could be used for this evaluation. In an automated approach to clustering, like with Genes@Work [reference], every pattern would be considered, in order, until every tissue was placed in a cluster.

by Genes@Work's own method, as well as the highest support patterns found by Genes@Work. Finally, a list of genes most important to classifying this data will be compiled and compared against those from Teiresias.

### 3.2.7 Other Issues

In addition to the above executions other issues will be considered when comparing Teiresias, Splash and Genes@Work. The ease of use is an important issue for Biologists when they use utilities developed by computer scientists. A graphical user interface (GUI) provides a biologist with a familiar windows-like interface but is more restrictive than a command prompt. Thus issues of utility must also come under consideration. Both Splash and Teiresias provide other parameters to either trim the reported patterns further, or reorder them in some way and these parameters may or may not be useful for pattern discovery on microarray data. Even specifying the equivalence classes is handled differently in Splash and in Teiresias.

Also, when evaluating results the data requirements of each binning method must be taken into consideration when examining results. Clearly, the training-data technique requires more data than the other methods.

### 3.2.8 Cluster

For each of the above data two data sets the program "Cluster" will be used to form a hierarchical clustering of the input. The patterns found in the above data will be compared to the results from Cluster. That is, either Cluster will confirm the results from these other techniques, or these other techniques will find different results than Cluster.

For the yeast cell-cycle data, the gene-list from the highest support patterns with well defined biological roles (the "best" patterns) will be checked against the results from Cluster to see if the same genes are closely correlated. For each pattern, the smallest group that contains all (or nearly all) the genes will be considered the equivalent grouping in Cluster. For each equivalent group, the correlation value of the group, and the number of genes it contains will be reported. The Cluster group will almost certainly contain

more genes than the pattern. These "extra" genes will be evaluated on how well they agree with the biological significance associated with the pattern. If the Cluster grouping contains proportionally more genes that agree with the core-function of the genes in the pattern, the Cluster group will be considered better. If the Cluster group contains genes that dilute the core-function of the pattern, then the results will be considered worse.

For the lymphoma data set, the classification made by Cluster will be considered correct. The longest patterns for various levels of support, will be evaluated against the Cluster results based on how well they agree.

### *3.3 Materials*

All tests of the various applications will be conducted on either a SunBlade 1000 (SOLARIS 5.8) or a Beowulf Cluster of 33 Pentium-3 866MHz (256KB L2 cache) with 0.75GB of memory each. The SunBlade has 4 GB of RAM and a single processor running at 600MHz with a 4MB external cache.

## 4 Results

### *4.1 Yeast Cell-cycle*

The yeast cell-cycle data was used for two purposes. First, it was used to benchmark time and memory usage of Splash, Teiresias, and Genes@Work. Second, it was used to evaluate the utility of 3 different binning techniques, '+/-/0', pre-set value, and K-means. In all cases, unless otherwise noted, the original data set of ~6,000 genes was filtered to 612 genes using a variational filter. However, it became apparent that Genes@Work was not able to finish execution on such a large data set with the same pattern density requirements as Teiresias, and more alarmingly, it appeared that Splash did not find the same patterns as Teiresias. The latter problem is termed "the missing pattern problem" and is outlined below. However, by executing exhaustive pattern discovery on a smaller data set, the missing pattern problem is over-come, and the three applications can be

successfully benchmarked. Finally the results from each of the 3 binning techniques, using the full 612 genes with Teiresias as the pattern discovery application, is presented.

### 4.1.1 Missing Pattern Problem

After executing pattern discovery on the '+/-/0' data set, it became apparent that Splash only found a subset of the patterns found by Teiresias. Although the reason for this is not known, the following example illustrates the effect concretely. With the following input file for both Teiresias and Splash and parameters *Z*=4, *K*=2, *L*=4, *W*=9:

```
>Stream 1
ECAAAFCCACGDCADHCACAICDDAKCACALDAACMACACNACADOACACPAADCQDDDARACACSACCATC
ACAVACACWACACXACACYDCAAE
>Stream 2
ECCAAFCDACGDCADHACCDICCDAKAACCLCDACMACACNACACODCACPACACQDCDDRACDDSADDATC
CCAVCAACWADACXACAAYDACCE
>Stream 3
EACACFAADDGDCADHCCACIACACKDACALDCACMACACNACACOACACPADACQACACRACCCSACDATC
ACCVCCDAWACACXDAAAYCCAAE
>Stream 4
ECCAAFCACCGCCADHDACDICAAAKCDCALCDACMACACNACACOACACPADACQDDCCRACDASDDAATC
DDDVCCAAWCCACXDCAAYCCAAE
```

Both algorithms find many patterns in this data. However, Splash misses the following pattern that Teiresias finds:

```
I..A.K..CAL..ACMACACNACACOACACPADACQ...CRAC..S...ATC...VCC.AW.CACXD.AAYC
CAAE    (Stream 3, position 20) (Stream 4, position 20)
```

Visual inspection confirms that this pattern does exist in the data set and it is maximal (with *L*=4 and *W*=9). Although Splash misses this one pattern, it does find the following 3 patterns which involve streams 3 and 4 (the relevance of which will become clear):

```
Pattern 1:
ECCAAFC..CG.CADH..CDIC..AK..C.LCDACMACACNACACO.CACPA.ACQD...RACD.S.D.ATC
...VC.A.W..ACX.CAAY....E  (Stream 2, position 0) (Stream 4, position 0)

Pattern 2:
```

```
K..C.L..ACMACACNACACO.CACPA.ACQ....RAC..S...ATC...VC...W (Stream 2,
position 25) ( Stream 3, position 25) ( Stream 4, position 25)

Pattern3:
E.CA.F....GDCADH.C..I.C..K.AC.L..ACMACACNACACO.CACPA.ACQ.C..RAC..SA.DATC
.C.VC...WA.ACX..AAY....E ( Stream 2,  position 0) ( Stream 3,  position
0)
```

Here it can be seen that Splash discovers patterns between streams {2, 4}, {2, 3, 4}, and {2, 3}. However, Splash does not discover a pattern between streams 3 and 4 only. Notice, that pattern 2 is the parent of patterns 1 and 3. That is, by extending pattern 2 to the left and right, as well as dereferencing a few "don't care" characters, patterns 1 and 3 can be formed. Observe also that pattern 2 is similar to the missing pattern except that it has not been extended to the left or right. That is, pattern 2 is also the parent of the missing pattern. Therefore it is possible that the missing pattern is absent because pattern 2 was not extended correctly.

To investigate further, Splash and Teiresias were executed with the following input, which is identical to the above except the first 20 characters of each stream has been removed:

```
>Stream 1
ICDDAKCACALDAACMACACNACADOACACPAADCQDDDARACACSACCATCACAVACACWACACXACACYD
CAAE
> Stream 2
ICCDAKAACCLCDACMACACNACACODCACPACACQDCDDRACDDSADDATCCCAVCAACWADACXACAAYD
ACCE
> Stream 3
IACACKDACALDCACMACACNACACOACACPADACQACACRACCCSACDATCACCVCCDAWACACXDAAAYC
CAAE
>Stream 4
ICAAAKCDCALCDACMACACNACACOACACPADACQDDCCRACDASDDAATCDDDVCCAAWCCACXDCAAYC
CAAE
```

With this input, the missing pattern from above is now found by Splash. This would suggest that Splash may be able to discover the pattern, but the pattern is lost when the input streams are extended to the left. However, this effect is not seen every time a

pattern can be extended to the left otherwise many patterns should be missing in any output produced by Splash. Attempts to replicate this behavior with other, shorter, artificial input-streams have not shown this result. The reason for this effect is unknown, but it does appear to be a genuine error in Splash. Further investigation would likely require Splash's source code and a great deal of time to determine the exact cause of this fault.

The missing-pattern problem effectively makes it impossible to benchmark Splash against the other algorithms because the result set would not include all the patterns. Fortunately, however, this problem does not occur if pattern discovery parameters are set such that all aligned patterns are discovered ($L$=2, $W$=$Z$+2). However, discovering every aligned pattern is a very computationally expensive endeavor. In order to do it, the input data sets must be very small.

### 4.1.2 Time and Space Usage

Teiresias, Splash and Genes@Work were used to discover patterns in each of the 4 data sets (representing the first 25, 30, 35 or 40 streams from the data used in Section 4.1.4) with $K$ ranging from 2 to 5. The memory usage, execution time[11] and number of patterns found for each execution is presented in Table 4.1. In Table 4.2 the number of aligned and total patterns (both aligned and unaligned) found by each program is shown except for Genes@Work which only finds aligned patterns. Genes@Work was unable to finish pattern discovery on the largest data set (40 streams) within 24 hours.

[11] Elapsed or "Wall clock time"

**Table 4.1. Execution time and Memory usage for Teiresias, Splash and Genes@Work**

| Streams | *K* | Teiresias | | Splash | | Genes@Work | | # of Aligned Patterns |
|---|---|---|---|---|---|---|---|---|
| | | Time (s) | Mem. (Mb) | Time (s) | Mem. (Mb) | Time (s) | Mem. (Mb) | |
| 25 | 2 | 118 | 41 | 124 | 16 | 1354 | 122 | 32783 |
| | 3 | 99 | 42 | 102 | 16 | 1433 | 118 | 32483 |
| | 4 | 73 | 35 | 76 | 15 | 1214 | 103 | 30343 |
| | 5 | 54 | 27 | 45 | 14 | 989 | 99 | 24115 |
| 30 | 2 | 204 | 58 | 391 | 23 | 10354 | 133 | 64027 |
| | 3 | 174 | 54 | 344 | 23 | 9943 | 145 | 63592 |
| | 4 | 135 | 48 | 266 | 22 | 9654 | 149 | 59841 |
| | 5 | 100 | 40 | 179 | 20 | 7812 | 131 | 47797 |
| 35 | 2 | 350 | 87 | 1183 | 37 | 64159 | 190 | 123547 |
| | 3 | 302 | 80 | 1100 | 37 | 70879 | 190 | 122952 |
| | 4 | 233 | 69 | 906 | 36 | 73326 | 148 | 117056 |
| | 5 | 174 | 53 | 627 | 32 | 67177 | 150 | 96281 |
| 40 | 2 | 607 | 128 | 3990 | 66 | * | * | 233515 |
| | 3 | 515 | 125 | 3840 | 66 | * | * | 232735 |
| | 4 | 394 | 103 | 3357 | 64 | * | * | 223847 |
| | 5 | 304 | 85 | 2450 | 57 | * | * | 189355 |

* = did not finish under 24 hours

**Table 4.2. Number of Patterns discovered by Teiresias, Splash and Genes@Work**

| Streams | *K* | Teiresias | | Splash | | Genes@Work |
|---|---|---|---|---|---|---|
| | | Aligned Patterns | Total Patterns | Aligned Patterns | Total Patterns | Aligned Patterns |
| 25 | 2 | 32783 | 208442 | 32783 | 209971 | 19724 |
| | 3 | 32483 | 187224 | 32483 | 188575 | 19721 |
| | 4 | 30343 | 139142 | 30343 | 140021 | 19374 |
| | 5 | 24115 | 98398 | 24115 | 98927 | 17496 |
| 30 | 2 | 64027 | 317115 | 64027 | 319478 | 38226 |
| | 3 | 63592 | 289548 | 63592 | 291677 | 38224 |
| | 4 | 59841 | 221359 | 59841 | 222785 | 37722 |
| | 5 | 47797 | 159409 | 47797 | 160243 | 34536 |
| 35 | 2 | 123547 | 483375 | 123547 | 486764 | 74519 |
| | 3 | 122952 | 446765 | 122952 | 449830 | 74516 |
| | 4 | 117056 | 349625 | 117056 | 351690 | 72827 |
| | 5 | 96281 | 256551 | 96281 | 257800 | 68749 |
| 40 | 2 | 233515 | 720116 | 233515 | 724785 | * |
| | 3 | 232735 | 674669 | 232735 | 678926 | * |
| | 4 | 223847 | 545797 | 223847 | 548664 | * |
| | 5 | 189355 | 413867 | 189355 | 415645 | * |

* = did not finish in 24 hours

There are a number of important conclusions that can be drawn from the data in Table 4.1 with regards to execution time, memory usage, and how this memory usage may reveal implementation decisions on the part of the authors. These conclusions are discussed in the following four paragraphs.

First, except for very short runs (e.g. *K*=5, *Streams* = 25) Teiresias is faster than Splash. Graphically this can be seen in Figure 4.1. As the number of streams increases, this speed difference becomes more pronounced: From approximately equal run time for *K*=2,

Streams = 25, to a 100% increase in time for *K*=2 Streams = 30, to a 200% increase for *K*=2 Streams = 35 and a five-fold increase for *K*=2 Streams = 40. Likewise, Splash is significantly (10x – 100x) faster than Genes@Work. There is a caveat, however, to concluding that Genes@Work is 10 to 100 times slower than Splash; because Genes@Work is written in Java its execution time depends heavily on the Virtual Machine (VM) that is used to run it. It is known, for instance, that running Genes@Work using the java option *-server* that execution times decrease by ~30%. However, *-server* is not the default VM for most people, and the batch files provided with Genes@Work do not specify the *-server* option when invoking the VM. Thus it was decided that executions of Genes@Work should be conducted with the default VM even though it results in worse execution time.

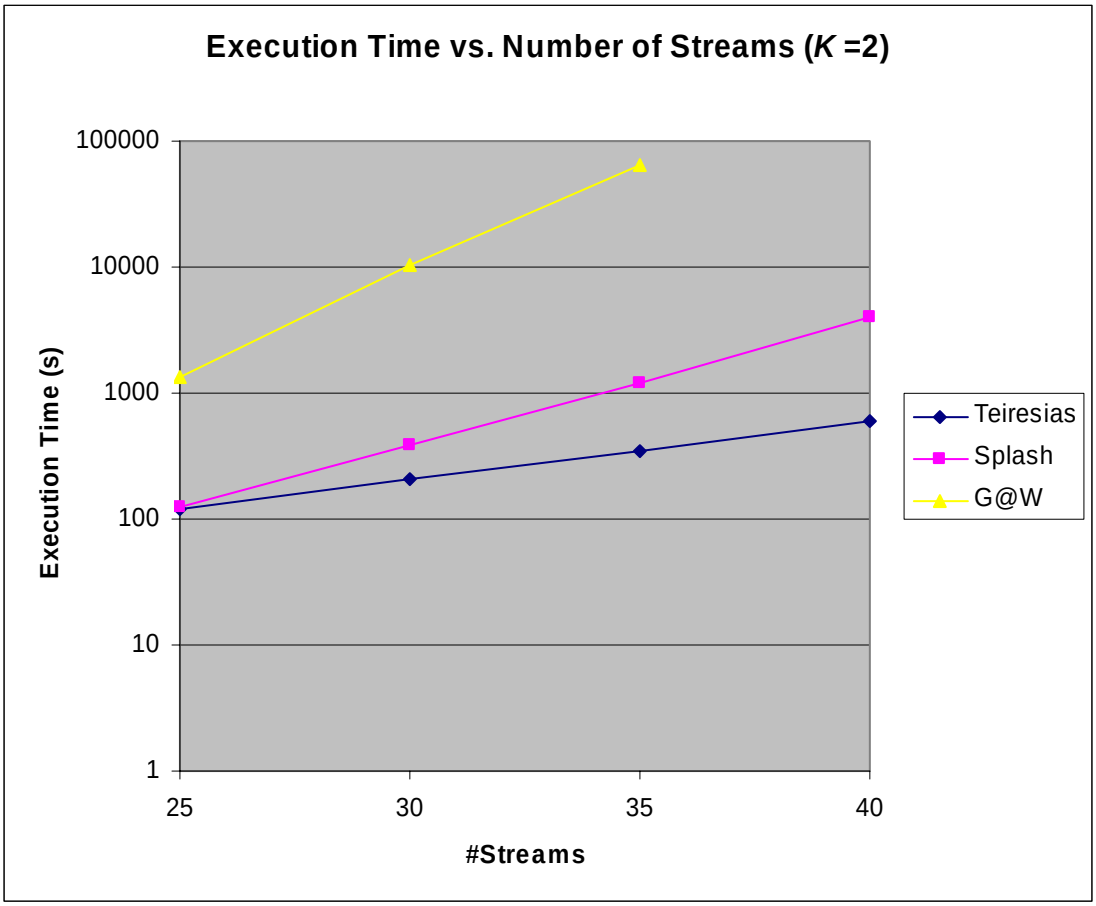


**Figure 4.1 Log execution time of Teiresias, Splash and Genes@Work vs. Number of streams (with constant support, *K*=2) , with linear interpolation of the data points**

Second, without exception, Splash uses (2x – 3x) less memory than Teiresias and Teiresias uses (3x – 4x) less memory than Genes@Work. This can be seen in Figure 4.2 where the memory requirements for pattern discovery on 25, 30, 35 and 40 data streams with $K$=2…5 is shown. With regards to Genes@Work's inefficiency, it is important to note that, in addition to pattern discovery, the Genes@Work application includes many other features (a GUI, hierarchical clustering) that increase the memory requirements of the application at start-up. However, this start-up cost is constant and does not change with different input data sets. Thus, doubling the input data size does not double the memory requirement. This effect is evident from Table 4.1 where going from $K$=2 Streams = 25 to $K$=2 Streams = 35, Teiresias uses 2x more memory (41Mb to 87Mb, a difference of 46Mb) for pattern discovery where as Genes@Work uses only ~50% more memory (122Mb to 190Mb, a difference of 78Mb). Graphically, this can be seen by observing the slope of the trend lines for each application in Figure 4.2. Although Genes@Work has a greater slope than Teiresias, it is not 4-times greater, even though looking at the raw numbers would imply Genes@Work uses 4x as much memory. In Figure 4.2 we can also see that not only does Splash use less memory than Teiresias, but as the number of patterns discovered increases, Splash also uses a smaller and smaller percentage of the amount of memory required by Teiresias.

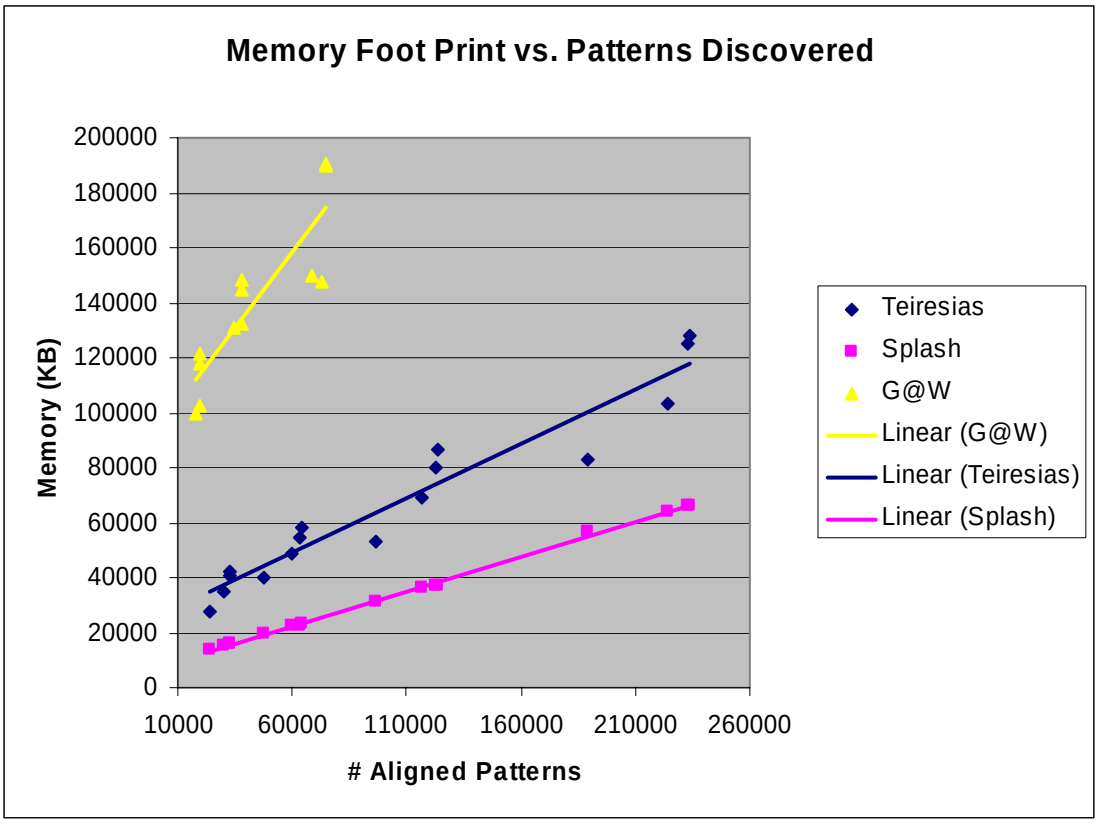


**Figure 4.2. Memory used vs. the number of patterns discovered for Teiresias, Splash and Genes@Work for 25, 30, 35 and 40 data streams and *K*=2…5, with linear interpolation of the data points**

Another interesting point to note is that Teiresias uses less memory per pattern discovered when *K* is high, than when *K* is low. To see this effect in Figure 4.2, first notice that Teiresias' data points occur in 3 clusters of 4 points each. Each cluster corresponds to the number of streams in the data set used (25, 30 or 35) and each cluster, individually, appears to conform to an exponential curve. In each exponential curve, the highest point is for the lowest value of *K* (*K*=2) and the lowest point for *K*=5. It can then be seen, to discover ~100,000 patterns (*K*=5, streams = 35) takes less memory than to discover ~68,000 patterns (*K*=2, Streams = 30). Practically, this means that pattern discovery with a large data set and high values of *K* will be more memory efficient than lower values of *K*, even with some slightly smaller data set.

Third, the way in which Splash and Teiresias use memory reveals potential implementation details. It is known from its author that Teiresias uses a hash-table to store intermediate patterns [I. Rigoutsos, personal communication, Jan. 21st, 2003]. Although no details are provided on how Splash stores intermediate patterns, it is known that pattern discovery lends itself to a tree-like data structure (see Section 2.5.5) and thus a tree could be used to store the intermediate patterns. If Splash does use a treelike structure (or any other non-hashing structure) to hold intermediate patterns it would partially explain why Teiresias uses more memory than Splash. Hash-tables are fast, but inefficient data storage structures; Splash, using a tree-like structure would be slower but more efficient. This might also explain why Teiresias will occasionally require more memory to discover fewer patterns as outlined above. In addition, because Teiresias must keep both a prefix-wise and suffix-wise list of all the patterns, it will require more memory than Splash to run. Lastly, because Splash only allows 22 characters as valid input, it can represent patterns using only 5 bits ($2^5 = 32$) per character and thus saves on Teiresias' requirement of 8 bits per character.

Lastly, it can be seen in Figure 4.2 that Genes@Work has some erratic behavior with respect to its resource management. Like Teiresias, it too occasionally requires more memory to discover fewer patterns and occasionally requires more time to discover fewer patterns. The reason for this is unknown, but is in all likelihood due to the VM and the way in which it optimizes code at runtime and allocates memory.

### 4.1.3 Pattern Properties in a subset of the YCC data

Although the data collected above is useful for time and space analysis there is also information to be discovered about the patterns themselves which helps illustrate the efficiency of each program and reveals a possible way in which to select patterns for further analysis. From this data we can determine the ratio of aligned patterns to all patterns discovered. This is a measure of the useful work done by each application. If very few of patterns discovered are aligned (the only patterns used in this work) then the application has wasted process time and memory on discovering unaligned patterns.

Further, the way in which the number and length of aligned patterns change with increasing support, *K*, is examined and a method of selecting important patterns is shown.

In Table 4.2, we see a list of the number of patterns discovered by Teiresias, Splash and Genes@Work. We can use the number of patterns discovered as a measure of the useful work done by each program. There are 2 major points to be drawn from this data. The first and most obvious result is that even with break characters, aligned patterns represent a small portion (15-25%) of the total patterns discovered. This means that Teiresias and Splash are doing a great deal of unnecessary work because 75% of the discovered patterns are not useful to this analysis. The number of discovered unaligned patterns could be reduced using a lower value of *Z*, at the cost of increasing the total stream length and reusing many characters as break-characters. However, a better solution would be for Splash and Teiresias to have an added function of being able to find only aligned patterns such as an *–alignedonly* parameter. Second, Genes@Work finds significantly fewer aligned patterns than either Teiresias or Splash. This is because Genes@Work incorporates a feature whereby it attempts to remove correlated patterns. Correlated patterns are described below in 4.1.4. The exact methodology that Genes@Work uses for determining a correlated pattern is not known. However, it is believed that Genes@Work does find all the aligned patterns that Teiresias and Splash find, but simply removes the correlated patterns from the final results.

Table 4.3 shows some interesting properties of the yeast cell-cycle data set. First, the number of aligned patterns discovered (for any value of *K*), as expected, increases exponentially with respect to the number of streams in the data set. Graphically this can be seen in Figure 4.3 , where the number of patterns discovered by Teiresias for K=2 for 25, 30, 35 and 40 streams is shown.

**Table 4.3. Number of aligned patterns of given length for varying support**

| #Literals | Support of at least *K* | | | | #Literals | Support of at least *K* | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *K*=2 | *K*=3 | *K*=4 | *K*=5 | | *K*=2 | *K*=3 | *K*=4 | *K*=5 |
| 1 | 0 | 0 | 0 | 0 | 29 | 30 | 53 | 1 | 0 |
| 2 | 0 | 0 | 16 | 70 | 30 | 28 | 33 | 0 | 0 |
| 3 | 0 | 0 | 171 | 710 | 31 | 40 | 28 | 0 | 0 |
| 4 | 0 | 15 | 654 | 2263 | 32 | 51 | 6 | 0 | 0 |
| 5 | 0 | 53 | 1440 | 3516 | 33 | 41 | 8 | 0 | 0 |
| 6 | 0 | 113 | 2017 | 4039 | 34 | 28 | 3 | 0 | 0 |
| 7 | 0 | 216 | 2266 | 4694 | 35 | 38 | 3 | 0 | 0 |
| 8 | 0 | 310 | 2623 | 5375 | 36 | 39 | 0 | 0 | 0 |
| 9 | 0 | 365 | 2814 | 5593 | 37 | 44 | 1 | 0 | 0 |
| 10 | 0 | 423 | 2972 | 5627 | 38 | 35 | 0 | 0 | 0 |
| 11 | 0 | 459 | 2950 | 4997 | 39 | 26 | 2 | 0 | 0 |
| 12 | 0 | 535 | 3002 | 4355 | 40 | 26 | 0 | 0 | 0 |
| 13 | 1 | 569 | 2907 | 3480 | 41 | 17 | 0 | 0 | 0 |
| 14 | 1 | 560 | 2457 | 2455 | 42 | 14 | 0 | 0 | 0 |
| 15 | 3 | 568 | 2124 | 1689 | 43 | 15 | 0 | 0 | 0 |
| 16 | 2 | 562 | 1730 | 1037 | 44 | 8 | 0 | 0 | 0 |
| 17 | 8 | 570 | 1312 | 724 | 45 | 7 | 0 | 0 | 0 |
| 18 | 7 | 512 | 1059 | 442 | 46 | 4 | 0 | 0 | 0 |
| 19 | 10 | 537 | 705 | 264 | 47 | 2 | 0 | 0 | 0 |
| 20 | 10 | 449 | 513 | 137 | 48 | 1 | 0 | 0 | 0 |
| 21 | 13 | 427 | 319 | 59 | 49 | 3 | 0 | 0 | 0 |
| 22 | 18 | 374 | 211 | 26 | 50 | 2 | 0 | 0 | 0 |
| 23 | 22 | 308 | 120 | 13 | 51 | 0 | 0 | 0 | 0 |
| 24 | 29 | 239 | 63 | 0 | 52 | 1 | 0 | 0 | 0 |
| 25 | 30 | 230 | 29 | 0 | 53 | 1 | 0 | 0 | 0 |
| 26 | 33 | 153 | 12 | 0 | 54 | 0 | 0 | 0 | 0 |
| 27 | 43 | 116 | 4 | 0 | 55 | 0 | 0 | 0 | 0 |
| 28 | 48 | 88 | 1 | 0 | 56 | 1 | 0 | 0 | 0 |

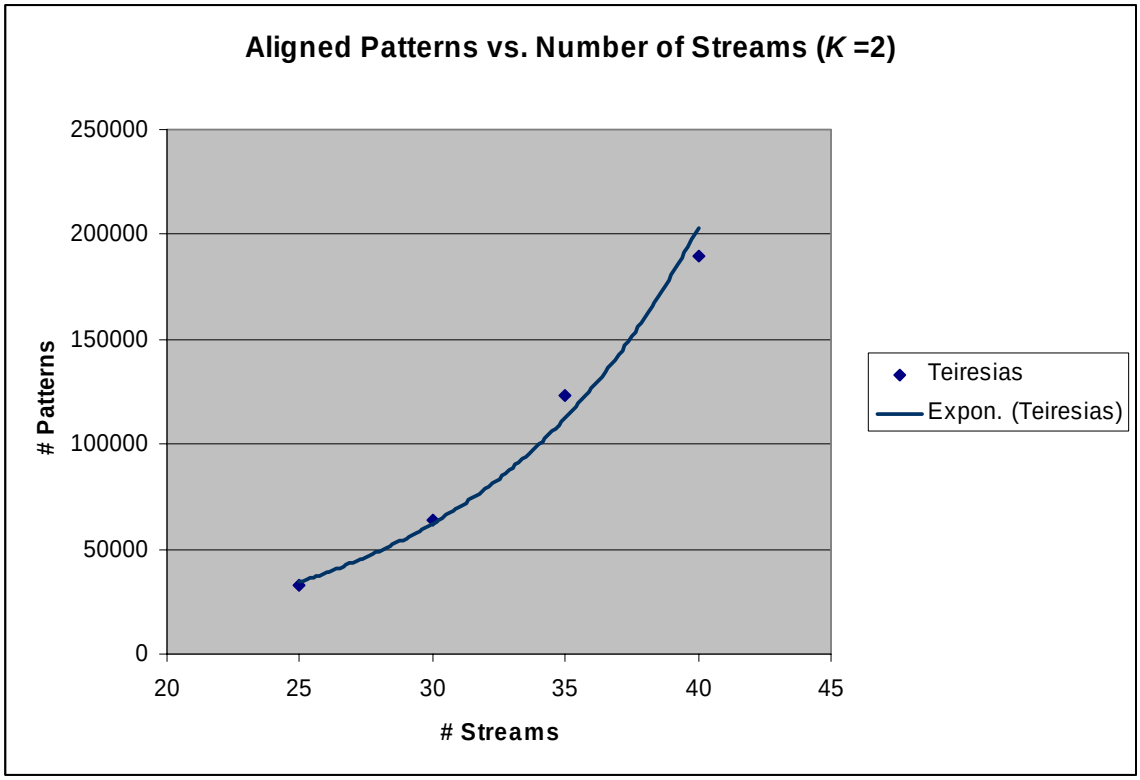


**Figure 4.3 Aligned patterns discovered by Teiresias vs. the number of streams in the data set to show exponential relationship, shown as a solid line**

As will be seen in subsequent sections, balancing the support and length of a pattern is very important in determining which patterns are interesting and likely useful for clustering. For example, a pattern with support 2 which contains 20 literals may be common and may occur at random. However, a pattern with 20 literals and support 20 may be very rare and thus is interesting and likely shows a real biologically significant relationship. Determining the greatest length of a pattern that occurs for some particular *K* may be a way of determining their importance. For instance, looking at Table 4.3, it may be concluded that patterns with support of exactly 2 and containing at least 40 literals may be interesting. We use the cut-off of 40 literals because when *K*=3 there are some patterns that occur with 39 literals. Generally, for any value of support *X*, the minimum number of literals required would be one more than the number of literals in the longest pattern supported by *X*+1 streams. The reasoning behind this approach is that if a pattern of any particular length occurs at a higher support, then it is more likely to be interesting at the higher support than at the lower. This does not mean that the pattern is

not biologically significant at the lower level of support, however. So, this technique favors to minimize the false-positive rate of patterns considered interesting. Although this intuitively seems like a reasonable approach, further investigation would be required to confirm this technique's usefulness. The technique used in the remainder of this work involves selecting the top 5 or 10 longest patterns for each level of support, which is a more conservative variation of the technique described here.

### 4.1.4 Sign-of-the-derivative technique

As mentioned in Section 2.4.2 this technique maps gene expression microarray data to +/-/0 based on whether the level of gene expression increases, decreases, or remains the same between two time-points. The amount of difference between two points in order for them to be considered different is defined ε. Of the original ~6,000 genes, the 612 genes with the highest variance were selected for pattern discovery. From this, 3 different data sets were constructed with ε values of 0.1, 0.2, and 0.4.

Because timing analysis already exists for the three pattern discovery algorithms and it is known that Teiresias and Splash either produce the same results (for exhaustive pattern discovery) or Splash produces a subset of Teiresias's results (for non-exhaustive pattern discovery), only Teiresias needs to be executed with this data. Further, from preliminary results, it was known that only Teiresias could finish execution on this data set using $L$=4, $W$=9, $K$=2. However, there are two aspects to Genes@Work that make it very unique from Teiresias. First, it implements a global density requirement, whereas Teiresias (and Splash) use local density requirements. Second, it has its own method of determining important patterns, or rather, a way of calculating the probability of a pattern. To investigate the effects of these properties Genes@Work was also used to discover patterns in this data set.

For each data set pattern discovery was executed using $L$=4 $W$=9 (for Teiresias) – meaning at minimum 4 literals must be present in any 9 character span in the pattern – and with support values $K$=2. In the following analysis patterns with a minimum support of 3 and 6 are examined, but executions with higher values of $K$ are not needed because those same patterns will be present in $K$=2. However, if for some values of ε Teiresias was unable to finish with $K$=2, then executions with $K$=3 and $K$=6 were attempted. Genes@Work was executed using the same three input data sets but a single density requirement of 30 literals[12] across the length of the data streams. Following this, the results of equivalent executions between pattern discovery applications were compared to confirm that they produced the same patterns. However, instead of examining every pattern, the focus was narrowed to the 20 most interesting patterns. Further, the most interesting patterns are evaluated for biological significance using GO annotation [GO].

#### 4.1.4.1 General Observations

Before exploring specific results of pattern discovery, a few general observations will be noted first along with a description of problems incurred when Genes@Work and the workarounds used to alleviate these problems.

As expected from Section 4.1.2, Teiresias runs faster than Genes@Work, which was unable to finish pattern discovery using a low density requirement. Therefore, Genes@Work was executed on the same data sets but with a higher literal density requirement.

As described above, after execution, it was seen if Genes@Work was capable of producing the same 20 longest patterns as Teiresias and vice versa.

The last and most important note is that very often the longest patterns are all supported by the same set of genes. As a result, if the longest pattern occurred between genes W, X

[12] This value was chosen because it takes at least 30 characters to form a pattern that spans the entire length of a 77 character data stream. It was also known from preliminary executions of Teiresias that the longest patterns almost always spanned the entire length of the input streams.

and Y then the next longest might occur between W, X and Z and the next longest between X, Y and Z and then W, Y and Z and so on. This occurs because there is a long pattern that occurs between all 4 genes and slightly longer, more specific patterns between subgroups of the 4 genes. These pattern are said to be correlated. It is believed that Genes@Work attempts to remove some of these patterns after pattern discovery. However, as will be shown below, Genes@Work often fails to remove a number of these correlated patterns. The high occurrence rate of correlated patterns highlights a flaw in defining the longest patterns as the most interesting. Because a pattern supported by 4 genes is likely more biologically significant than a longer pattern supported by less genes. The solution to this problem is to use a scoring system that balances the length of patterns with their support (see Section 4.1.3). However, devising such a system is difficult because for any particular data set long patterns may give better biological results than higher support pattern or *vice versa*. This problem is exacerbated by the fact that a gene's expression is correlated to the expression of other genes (this is effectively what is being determined by microarray analysis) which makes determining the probability of a number of genes co-expressing difficult.

#### 4.1.4.2 Execution Times and Memory Usage for Pattern Discovery

The execution times and memory usage of Teiresias and Genes@Work are listed in Table 4.4. The first time listed for Genes@Work is the time taken to execute pattern discovery, and the second is the time taken to discover and remove correlated patterns. It can be seen that when Genes@Work uses more memory (and discovers more patterns) that the difference between Time1 and Time2 becomes significant. The rows in bold represent the executions where Teiresias, Splash and Genes@Work all finished pattern discovery.

**Table 4.4. The execution time and memory usage of Teiresias and Genes@Work for each value of ε and *K* for sign-of-the-derivative data binning technique.**

| ε | *K* | Teiresias | | Genes@Work | | |
|---|---|---|---|---|---|---|
| | | **Time(s)** | **Mem. (GB)** | **Time1(s)** | **Time2(s)** | **Mem (MB)** |
| 1 | 2 | 25572 | 4.2* | 6360 | 48420 | 249 |
| 1 | 3 | 24912 | 4.2* | 4260 | 32880 | 171 |
| **1** | **6** | **22557** | **3.9** | **742** | **742** | **85** |
| **2** | **2** | **16659** | **3.5** | **697** | **3110** | **134** |
| 4 | 2 | 27201 | 4.2* | DNF | DNF | DNF |
| 4 | 3 | 24973 | 4.2* | DNF | DNF | DNF |
| 4 | 6 | 29252 | 4.2* | 2970 | 2970 | 87 |

* indicates that execution did not finish due to memory usage

DNF indicates that execution did not finish in <24 hours

In terms of the number of patterns found for each execution, Teiresias finds between 11 million and 24 million patterns. Genes@Work finds many fewer patterns, from as little as 2 to a maximum of 90,000. This is due to two causes. First and most importantly, the global density constraint of 30 literals and second, the removal of patterns Genes@Work considers correlated.

As can be seen in Table 4.4, the fastest execution times occurred on data sets where ε=0.2, followed by ε=0.1 and ε=0.4. This is because when ε=0.2, the frequency of each data character (+/-/0) is roughly equal. When ε=0.1 there are very few '0' characters and when ε=0.4 the majority of the characters are '0' thus increasing the number of patterns (and their length). Although finding an ε value that makes the frequency of all characters exactly equal would minimize execution time, it is difficult to justify this approach biologically.

Because of the different density requirements placed on Genes@Work and Teiresias, it is important to examine how well the results from the three applications agree with each

other. Two applications are said to have found the same aligned pattern if the support list for that pattern is the same for both applications. This is a valid way of determining that two patterns are the same because there can be only one maximal aligned pattern that is supported by any unique set of genes. This method is used because it is faster than checking every literal in a pattern. In all cases Teiresias discovered the same 20 most significant patterns found by Genes@Work. However, Genes@Work generally failed to find most of the patterns found by Teiresias.

When considering the results from Genes@Work we must take into consideration two things. First, for the case $\varepsilon=2$ $K=6$ the shortest top-20 pattern discovered by Teiresias has a length of 24 characters. This places the pattern below the density restriction placed on Genes@Work, and thus it is not surprising that Genes@Work does not find the top 20 patterns that Teiresias does for these parameters. However, for all other values of $\varepsilon$ and $K$ (where Teiresias could finish execution) the shortest top-20 pattern consists of more than 30 literals and therefore Genes@Work should be able to find all such patterns. Even so, Genes@Work only finds between 2 and 9 of the top 20 patterns found by Teiresias for any data set. This low hit-rate is believed to be due to Genes@Work removing low support, correlated patterns from its output. Thus, instead of having multiple patterns between genes {W,X,Y}, {X,Y,Z}, {W,Y,Z}, ..., it has a single pattern between {W,X,Y,Z} that may be shorter than the patterns with lower support. Unfortunately this is hard to check for without knowing the algorithm that Splash uses to remove correlated patterns. If this explanation is indeed true, then internally, Genes@Work likely does find all the same top-20 patterns as Teiresias. Unfortunately, it is not currently possible to stop Genes@Work from removing uncorrelated patterns after discovering all patterns.

#### 4.1.4.3 Biologically Significant Patterns

This section looks into the biological significance of the most interesting patterns as discovered by Teiresias (where interesting means long) and Genes@Work (where interesting means improbable, see Section 2.5.3) on sign-of-the-derivative binned data. For each dataset, from each of the two applications, the set of genes that support the top 5 most interesting patterns are investigated and presented in a table. However, often many of the top 5 patterns are highly correlated to one another (see below). In such a case the

correlated patterns are omitted from the table and thus, some tables can contain as little as a single pattern. However, if all 5 of the investigated patterns appeared to have biological significance, or were highly correlated then the next 5 patterns were additionally examined. The following 11 tables are organized from lowest values of ε and *K* to highest. For almost every value of ε and *K* there are two tables. The odd-numbered tables contain the most significant patterns by length – from executions by Teiresias – and the even-number tables contain the most significant patterns as measured by Genes@Work. Because Teiresias was unable to finish pattern discovery on ε=0.1, *K*=2 (Table 4.5), and ε=0.1, *K*=3 (Table 4.6) the longest patterns discovered by Genes@Work are used instead. Because both Genes@Work and Teiresias were unable to finish when ε=0.4, *K*=2 and ε=0.4, *K*=3 that data was omitted. In the case of ε=0.4, *K*=6, Teiresias was unable to finish and Genes@Work only found 2 uncorrelated patterns thus only 1 table is presented.

The table columns in this section are organized as follows. "Rank" refers to the rank of the pattern's significance with 1 being the longest (or least probable) pattern and 10 being the $10^{th}$ longest (or $10^{th}$ least probable) pattern. Because Teiresias' output includes the template of the discovered patterns, it is shown in the second column. The break characters have been removed since they make reading the template more difficult. Genes@Work, on the other hand, does not output the pattern's template, so the log-probability of the pattern is shown in the table instead. The last column lists the genes that support the pattern. When the gene's standard name is known, it is used; otherwise the systematic name is used.

In many cases the most interesting patterns are highly correlated to one another. In such instances the patterns are replaced by a single representative pattern. The representative pattern consists of the first pattern that contains the majority of the correlated genes and subsequent genes found in other, similar patterns are indicated by enclosure in curly braces. In all of the tables it can be seen that some ranks are missing. If a rank is missing it is for one of two reasons. One, the missing pattern was highly correlated to another pattern already in the table. In this case, the correlated pattern is removed and any genes

not appearing in the representative pattern are added to it (the representative pattern). Two, the pattern is supported only by genes labeled “dubious” or “hypothetical”.

Gene Ontology (GO) annotation [GO, ’03] was used to confirm the biological significance of each gene group. For each pattern, the genes that compose the pattern are submitted to the “GO term finder” (through a web interface) and a probability is assigned by the web service. The probability is presented here, where applicable, in the discussion of each table. Note, that this probability is different from the probability assigned to a pattern by Genes@Work and will be indicated by pre-fixing the probability with “$p$=”. This probability is calculated from the hypergeometric distribution and is a measure of how likely, at random, one would expect to find a particular group of genes that share some property. A lower probability means that the set of genes has more biological significance. A higher probability means that the set of genes is less biologically significant and more likely to have occurred at random. This probability is dependant on 4 values: 1) the number of genes that share some property (such as being involved in cell-cycle); 2) the number of genes that support the pattern (as reported by the pattern discovery program); 3) the number of genes that contain that property in the whole genome, and 4) the total number of genes in the whole genome. Calculating this probability is very similar to the probabilities of a scoring hand in poker. The probability of having a pair of aces in a poker hand is dependant on the following values: 1) there are 2 aces in your hand. 2) a poker hand has 5 cards. 3) there are 4 aces in the deck and 4) a deck consists of 52 cards. It should be noted that the probabilities presented here are likely lower than they should be for the data collected here. This is because SGD has annotations for ~7,700 genes, whereas the data was collected by Spellman was for only ~6,000 genes (this is the equivalent of having a smaller deck in poker). However, the probabilities serve as a guide to the likelihood of having selected these genes at random, and should be fairly comparable to one another. There is no accepted value for a maximum allowable probability for a group of biologically significant genes. So, the standard 95%-confidence interval ($p$=0.05) will be used.

**Table 4.5. Longest patterns discovered by Genes@Work for ε=0.1, *K*=2**

| Rank | Probability (log10) | Supporting Genes |
|---|---|---|
| 1 | -25 | CUP1-1 CUP1-2 |
| 2 | -47 | KCC4, MSH6, YCL022C |
| 3 | -47 | RFA1, RSR1, MSH2 {MSH6} |
| 4 | -45 | MRT4, EBP2, YMR269W |
| 6 | -45 | NOP6, MRT4, UTH1 {NOP7} |
| 7 | -10 | PRM1, SCW10 |
| 8 | -10 | PRM5, DIA1 |

In Table 4.5 we see the longest genes as discovered by Genes@Work with ε=0.1 and *K*=2 (Teiresias output was not used as Teiresias did not finish execution with these parameters). The rank 1 pattern is easy to verify as biologically significant since CUP1-1 and CUP1-2 are two copies of the same gene. This result suggest real patterns can be picked out by the sign-of-the-derivative method. The low listed probability of this (very biologically significant) pattern is due to its low support. The genes in the rank 2 pattern share little in common except they are all cell-cycle related genes ($p$=0.005). The rank 3 pattern is highly correlated to the rank 5 pattern (supported by RFA1, RSR1, and MSH6) – thus MSH6 is included in the rank 3 pattern, and the rank 5 pattern is omitted. RFA1, and MSH2&6 are all involved with DNA-dependant DNA-replication ($p$=5e-6). RSR1 (a gene involved in axial budding ) has less in common with the other genes except , generally, all these genes are involved with cell proliferation ($p$=4e-5). In the 4$^{th}$ pattern MRT4 and EBP2 are both involved in rRNA processing ($p$=0.00153), whereas the molecular function of YMR269W is unknown. In the 6$^{th}$ ranked pattern, which is correlated to the 9$^{th}$ and 10$^{th}$ patterns, all the genes are involved in cytoplasm organization and biogenesis ($p$=9e-5) and NOP6, NOP7 and MRT4 are specifically involved in ribosome biogenesis ($p$=4e-5). The 7$^{th}$ ranked pattern is supported by two genes that are both involved in cellular fusion during mating [SGD], but SGD does not group these two genes together and thus there is no probability provided. The last pattern is supported by another gene that aids in mating (PRM5) and a "dubious ORF", so it is not possible to comment on its bioloigcal significance.

**Table 4.6. Least probable patterns discovered by Genes@Work for ε=0.1 *K*=2**

| Rank | Probability (log10) | Supporting Genes |
|---|---|---|
| 1 | -134 | NOP7, HCA4, MRT4, EBP2, MRD1, YMR269W, YPL146C {DIM1, NOP6} |

In Table 4.6 there is only 1 pattern to consider because each of the next 9 patterns are correlated to this one and thus are removed. This pattern is from the same data as was used for generating Table 4.5, except here the least probable patterns are favored instead of the longest. All the genes that support this pattern (with the exception of YMR269W and YPL146C – whose function is unknown) are involved in rRNA processing ($p$=3e-11). Although little is known about YMR269W it is known that it is involved in cell growth and maintenance and is believed to be involved in protein synthesis [SGD]. The grouping in the above table would strongly support that conclusion and would almost certainly indicate that this gene is involved with rRNA processing.

**Table 4.7. Longest patterns discovered by Genes@Work for ε=0.1 *K*=3**

| Rank | Probability (log10) | Supporting Genes |
|---|---|---|
| 1 | -47 | RFA1, RSR1, MSH2 |
| 2 | -47 | KCC4, MSH6, YCL022C |

The content of Table 4.7 is very similar to that of Table 4.5. This is likely because Genes@Work removed many of the other support 2 patterns from the final output data set because it determined that these patterns were correlated with other, higher support patterns.

**Table 4.8. Least probable patterns discovered by Genes@Work for ε=0.1 *K*=3**

| Rank | Probability (log10) | Supporting Genes |
|---|---|---|
| 1 | -134 | NOP7, HCA4, MRT4, EBP2, MRD1, YMR269W, YPL146C {DIM1, NOP6} |
| 2 | -134 | MOB1, ACE2, SRC1, CDC5, HOF1 |

In the first line of Table 4.8 exhibits the same rRNA-processing group that occurred in Table 4.6. In the second row we also have a group of genes that seem to bare little functional resemblance (all are involved in cell proliferation $p$=3e-6). However, MOB1, CDC5 and HOF1 are all localized in the bud neck region ($p$=2e-5) and thus may share some functional relationship. Although ACE2 does not localize in the bud neck region, it is a transcriptional activator that mediates bud site selection, bud development, and cell separation and is known to interact with MOB2 [Bryce, '03]. It is therefore possible that ACE2 is part of a regulation cascade that indirectly regulates the other genes.

**Table 4.9. Longest patterns discovered by Teiresias for ε=0.1 *K*=6**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | +.-.-..++..-...++....--.+.---.-+++.+-.-.-...++-.+..-...+...+...-...++.....- | HTA1, HHT1, HTB1, HTA2, HHF2, HHT2 |
| 2 | +..+..+.--....+.---....+.-.-..++.+-.-.-..+.+-.-.+--..-.++.-....-..++...- | RFA1, KCC4, MCD1, MSH6, RSR1, YCL022C { SPT21, ASF1} |
| 5 | -+.-...+.+..--....++.--.+-+-+-.-..+.+-+-..+-.-....++..---.....- | MOB1, ACE2, CDC5, HOF1, YIL158W, YML119W |

In Table 4.9 we see one of the most common, highest support patterns for all of the yeast cell-cycle results. The rank 1 pattern here is supported by 6 histone genes ($p$=3e-9). It can be seen that this pattern is only supported by 33 literals (approximately half of the maximum). However, these few literals are clearly enough to form a biologically significant pattern. The rank 2 pattern (similar to the rank 1 pattern in Table 4.7) is

largely involved in cell proliferation (RFA1, KCC4, MCD1, MSH6, RSR1; $p$=6e-5) but other genes (KCC4, RSR1) are involved in axial budding ($p$=0.0002) and 3 others (RFA1, MSH6, ASF1) are involved in responding to DNA damage ($p$=0.0005). SPT21, which does not seem to share many properties with the rest of the group, is required for transcription of HTA2, HTB2, HHF2 and HHT2. Interestingly ASF1 binds to histones, so there may be biological significance to the co-regulation of SPT21 and ASF1. The third pattern (similar to the rank 2 pattern in Table 4.8) has all its supporting genes involved in cell proliferation ($p$=5e-5).

**Table 4.10. Least probable patterns discovered by Genes@Work for ε=0.1 *K*=6**

| Rank | Probability ($log_{10}$) | Genes |
|---|---|---|
| 1 | -137 | FIG1, PHO5, HXK1, PNC1, SPO12, RAD27, RMA1, SUN4, CHS1, YCRX18C, YER067W, YHR087W |
| 3 | -129 | CDC9, PCL9, RAD27, FUS2, SUN4, YDR149C, YER067W, YHR087W {SPO12, CHS1} |

The least probable pattern in this data set is supported by many genes. All these genes have little in common with one another. However, they seem to form subsets of functionally related groups. For instance, PNC1 and RAD27 are both involved in cell death ($p$=0.00035). PNC1, RAD27, FIG1 and CHS1 are all involved in cell development ($p$=0.002), whereas FIG1, PHO5 and SUN4 all are part of the cell wall ($p$=0.0003). There is very little biological evidence to support the grouping of genes which compose the second pattern shown here. CDC9, PCL9, RAD27, SPO12 are all involved in the cell-cycle but there is no more specific GO term that can be applied to them.

These results (and more below, see Table 4.12) highlight a problem with the Genes@Work scoring system when used with this data set. The system seems to overly favor high support rather than length of the pattern. If the scoring system was changed so that it would favor smaller, more highly related groups, their biological significance would be easier to confirm and the SGD probabilities would be better.

**Table 4.11. Longest patterns discovered by Teiresias for ε=0.2 *K*=2**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | 0+-+0000..+0.+.+..--0-++-00+0.+.+-+-+-+-+-+-+-+--+....0-+-+00..++--.-000000++ | CUP1-1, CUP1-2 |
| 5 | -.0+.-.+..+..0.0.-.0..+.---+-+-+-+-+-+-+-+-+-+-+-.+-+-+00+-..+000-+0++--.0..-0 | MRT4, EBP2 |
| 6 | ++-+++.+----0.++--..0--+.-.-...++0.---0-00+.0...+--..--+++-.-+.-0+.++00-+.-- | AXL2, CLN2 |
| 7 | -+.+--..00+..+-+-00+-+-.0.+-+-+-+-+.+-+-.+0-00.0.-+-+--+-+..+.+-.-+-....0-.+ | EFB1, SSA1 |
| 8 | +.-.-..++.--0-.++.0-.--.+0---.-+.+0+-0-0-..+++-+.00-.-0+.-++.+---+.++.+-.0- | HTB1, HHT2 |

The rank 1 and rank 5 patterns here are similar to the rank 1 and rank 4 patterns in Table 4.5. The rank 6 pattern consists of genes involved in reproduction ($p$=0.0005). The two genes in the rank 7 pattern are both a part of protein metabolism ($p$=0.04) and the last pattern are 2 of the 6 histones from Table 4.9 ($p$=1e-5).

**Table 4.12. Least probable patterns discovered by Genes@Work for ε=0.2 *K*=2**

| Rank | Probability (log10) | Supporting Genes |
|---|---|---|
| 1 | -128 | CDC9, PCL9, RAD27, FUS2, SUN4, CHS1, YCRX18C, YDR149C, YER067W, YHR087W {RMA1, HXK1} |

Here we see one pattern, where the next 4 patterns are highly correlated to the first. Except for CDC9, RAD27 (DNA repair/replication $p$=0.006) there is little else that associates these genes.

**Table 4.13. Longest patterns discovered by Teiresias for ε=0.2 *K*=3**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | +.-.-.+++..-.-.++0.----++0---.-+++0+-.-.-0-+++-++..-...+...+...-...++....0.- | HTB1, HTA1, HHF1 {HHT1} |
| 3 | -.0+.-.+..+..0.0.-....+.---+-+-+-+-+-+-+-+-+-+-+-.-...+..00....+..0-+.++.-.0..-0 | HCA4, MRT4, EBP2 |
| 4 | +.-...+.---...+.---.-0.+..--0.++.+-0-.-0.+.+-0-.+-----.+..-...+-.+++...-0.- | POL30, MCD1, MSH6 |

Here again we see how strong patterns appear consistently across different values for *K* and ε. The first two patterns here are smaller versions of the histone pattern (see Table 4.9) and the rRNA processing pattern (see Table 4.6). In the last pattern POL30 and MCD1 are both involved with DNA mismatch repair ($p$=1e-5) and all three genes are involved in DNA replication and the chromosome cycle ($p$=3e-5).

**Table 4.14. Least probable patterns discovered by Genes@Work for ε=0.2 *K*=3**

| Rank | Probability (log10) | Supporting Genes |
|---|---|---|
| 1 | -133 | YRO2, PHO88, MCD1, HSP42, RNR1, OLE1, SPO16, RMA1, MRP8, YHR095W |
| 2 | -129 | CDC9, PCL9, RAD27, FUS2, SUN4, CHS1, YCRX18C, YDR149C, YHR087W |

There is little to associate any of the genes appearing in the rank 1 pattern. Although, to be fair, little is known about YRO2, SPO16, RMA1 and YHR095W. The second pattern, like the rank 3 pattern in Table 4.10, involves genes with little known to associate them together.

**Table 4.15. Longest patterns discovered by Teiresias for ε=0.2 *K*=6**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | +.-.-..++..-.-.++....--.+.-.-.-+.+.+-.-.-..+++-+...-...+ | HHT1, HTB1, HTA1, HHF2, HHT2, HHO1 {HTA2, SVS1} |
| 2 | +..-....+..-...0.+...-...+.+-.-.-0.+.+-.-.+--..-.+..-...+-.++ | POL30, KCC4, MCD1, MSH6, MSH2, YCL022C |

In Table 15 we see the now familiar histone group with 2 additional genes, HHO1 (another histone associated gene) and SVS1. The function of SVS1 is largely unknown but cells with null mutations in this gene show increased sensitivity to vanadate. It is not clear why SVS1 is included in this gene group. However, it is known that vanadate oxoanions inhibit protein tyrosine phosphatases [Faure, '95] which effect CDC proteins, which in turn effect histones[Faure, '95]. In addition, vanadate compounds are known to directly bind and precipitate histones in solution [Michele, '97]. This may show a cell's desire to produce proteins protective against vanadate at a time when vanadate could cause the most harm. The second pattern in this table also contains many genes seen before. All are involved in the cell-cycle ($p$=1e-6) and all, except KCC4, are involved in DNA replication ($p$=5e-6).

**Table 4.16. Longest (and least probable) patterns discovered by Genes@Work for ε=0.4 *K*=6**

| Rank | Probability (log10) | Supporting Genes |
|---|---|---|
| 1 | -92 | HTA2, HHT1, HTB1, HTA1, HHF2, HH01, {HHT2} |

Genes@Work was the only algorithm able to complete pattern discovery on this data set with $K$=6. With a minimum requirement of 30 literals in the pattern only 2 patterns were found. The two patterns are correlated and the only difference between them was the presence of HHO1 and absence of HHT2, or vice versa. The single remaining pattern shown in Table 4.16 is the ubiquitous histone pattern.

From the above 11 tables 3 general conclusions can be drawn. First, pattern discovery in general and with the '+/-/0' technique in particular, is a valid and useful method of gene microarray analysis. With a few exceptions, the vast majority of the patterns listed above cluster together genes with similar function and thus can be considered biologically relevant. Many of the patterns that are not biologically relevant seem to be those favored by Genes@Work's scoring system: high support, low density patterns (see Tables 10, 12 and 14). Genes@Work's scoring system also does not favor biologically important patterns. For instance the histone group is not rated highly by Genes@Work.

The second generalization that can be made from these results is that strong patterns (patterns that are either long, or highly improbable) seem unaffected by changes in ε and thus appear in multiple tables. Consequently, the best way to pick ε may be to use the one that minimizes execution time by ensuring the '+', '-', '0' characters occur with equal frequency. The value of *K*, however, seems to have a stronger effect than ε on what patterns are ranked highly by Teiresias. Even then, a subset of the histone group appears with multiple values of *K*. *K*, not surprisingly, has a smaller effect on the results from Genes@Work. This is because the highly ranked (improbable) patterns often have support values greater than the minimum required support, and thus should be expected to appear approximately in the same order with many values of *K*. The strongest effect on the pattern rankings was, clearly, the use of two different scoring systems (either Genes@Work's system or ranking by pattern length). When given any multiplicity of patterns, Genes@Work and Teiresias never had similar patterns in the lower rankings. With the obvious exception of the rRNA processing pattern that was picked out by Genes@Work (see Table 4.6), the use of pattern length to determine significance seems to outperform Genes@Work's ranking system in terms of biological relevance. However, because the training data required by Genes@Work was artificial, the scoring system may have failed.

Third, the highest ranked patterns were almost always correlated to other highly ranked patterns. This was true for both patterns ranked by length (as discovered by Teiresias) and patterns ranked by probability. The only occasion when this did not appear to hold

was when the longest patterns discovered by Genes@Work were considered. It would appear that Genes@Work's removal of correlated patterns works well for the longest patterns and not as well for the least probable ones.

Regarding the methodology that was used, it is important to remember that the way in which the top patterns were selected and patterns with similar genes in their support list were merged was not automated. Clearly a system which would automatically merge lower support patterns that are supported by similar sets of genes (or better yet, a scoring system that weighs support vs. length in a biologically significant manner) would be preferable. This would also allow the number of patterns considered to be increased and consequently, find more clusters of correlated genes. Subsequently, a better over-all performance of the binning technique with regards to the biological significance of the patterns could be made.

#### 4.1.4.4 Biologically Significant Patterns in Pre-set Value Data

The pre-set value technique stratifies the microarray data into 5 bins based on the following delimiters {-1.00, -0.25, +0.25, +1.00} (see Section 2.4.4). In this case the 5 bins used are represented by the characters E…A, where A represents expression levels >= +1.00 and E represents expression levels <= -1.00.

This data set produced a large number of patterns and pattern discovery execution times became excessively long. This occurred primarily because certain ranges of values (-0.25 to +0.25) are more common than others, which causes more patterns to form. Because timing analysis data for Teiresias, Splash and Genes@Work was collected in a earlier stage of the methodology (see 4.1.2), the excessive execution times could be tolerated. The goal of this work is primarily to evaluate the utility of this binning technique. Because Splash and Teiresias found the same patterns and it is very difficult to implement this kind of binning in Genes@Work, only Teiresias is used for pattern discovery. However, even Teiresias was not able to complete pattern discovery using $L$=4, $W$=9, $K$=2 on this data set. In order to decrease execution times, highly similar data

streams were merged together into a single stream[13]. The data streams were compared in a pair-wise manner. If two streams share more than a certain percentage (in this case 90%) of characters in the same position, a single data stream is promoted as the archetype pattern and the other stream is removed from the data set. Doing this removed only 9 streams of data and had little impact on the execution times of pattern discovery. As a final attempt to complete pattern discovery on this data set Teiresias was executed on the Beowulf cluster, a collection of 32 computers over a high-speed network. Execution (wall clock) time on the cluster took 3 hours, 14 minutes and 20 seconds. Executing Teiresias with other values of *K* is not required as patterns with higher support values are included in this dataset.

Below are 5 tables of patterns discovered for *K*-values ranging from 2 to 5. As before, the longest patterns are considered the most interesting and the 5 to 10 longest patterns for each value of *K* were examined. When multiple patterns were supported by nearly the same genes (correlated patterns) they are represented by a single pattern in the table and the added genes are shown in curly braces. As before, the break characters have been removed for clarity. Lastly, for each pattern its biological significance is evaluated using SGD.

[13] This is favourable to increasing the value for *K*, which may result in missing lower support patterns which in the previous section had biological significance. Increasing *L* or decreasing *W* is also possible, but doing so forces the patterns to be too dense, and very few patterns occur across the entire length of the streams (data not shown).

**Table 4.17. The longest patterns discovered with *K*=2**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | AAEEEDAA....E...ABC.D.EAABCD.EEBAAAA..DED DCCCC.BA..CD.ECBB.C..EEECCBBBBBBB. | CLN2, YOL007C |
| 2 | .A..EEE.AAB..DDC..ABCDE.AABDEEE.DA.A.B.CDD E..BDCBCBD..D.EB.B.B.EEEDCBAA.BBBC | HHF1, HHF2 |
| 3 | AA.DEEED.A...D..BAA.C.EDAAACDEE..AAA..BCD E.DCBD.B..DCDDCBECBBB....D.BB.BBBBC | HTA1, HTA2 |
| 4 | AA.DEEE...BBDDECBAAACCE.A.A.DEEE.A.AABB. D.DD...EBB.DCDD.B..BBBCE..D.B..BBBBC | HTA2, HTB2 |
| 5 | AAEEEDAABB.DDDB.BC..DD.AA...EE..AAB.CCDD D.D.C.E.AB..DEEBAB..DD..ED.B.BB..CDD | MCD1, YOX1 |
| 6 | AAEEEDAA.CD.E.BAABCDD.EA....EEE.AAAAB.D... C.....AB..DDE.BBEC...EE..BB.BBBB.D | POL30, CLN2 |
| 7 | AA.DEEE..ABCCDDC..ABCD.DAAB..EEDD..A.BCC DD...BDCBCBD.DDB..D.CBC.E.DCB.AB.B.C | HHF2, HHT2 |
| 8 | AACDEEE..A.CCDD.BAABCD.DAA.CDEED...AB..C D..DCBD.BC.DCDD.BE...B.DE.DCBB.B.B.C | HTA1, HTT2 |
| 9 | AA...D.AB.DDDDBABC.CDD.AA.DEEED..AB.C.D.D CDBC...ABCDD..BA..C..D..D.B.BBCCCDD | MCD1, MSH6 |

In Table 4.17 we see the 9 longest patterns with a support of 2. The genes in the first pattern share nothing in common because YOL007C is uncharacterized (with possible role in chitin synthesis), whereas CLN2's role in is cell-cycle start. Patterns 2-4, 7 and 8 are all histone patterns (as seen previously) with each pattern individually having a *p*-value of 1e-5. Not all the clusters here have been identified before, however. Pattern 5 consists of two genes involved in the M-phase of mitosis ($p$=0.0003) and although the gene MCD1 has been encountered before (Table 4.9), YOX1 is a gene that was not highlighted by the sign-of-the-derivative technique. Pattern 6 consists of two more genes in a novel grouping. However, the closest similarity these genes share is that they are both expressed as a response to stimulus ($p$=0.0003). The final pattern, which agrees

with previously found patterns (see Table 4.9) are two genes involved in DNA replication and the chromosome cycle ($p$=0.001).

**Table 4.18. The longest patterns discovered with $K$=3**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | AAEEEDAA....E...ABC.D.EA.....EE.AAAA..D...C.....A. ..D.E.BB.C...EE..BB.BBBB | POL30, CLN2, YOL007C |
| 2 | AA.DEEE..A...D..BAA.C..DAA.CDEE....A...CD..DCB D.B..DCDD.BE...B....D.BB.B.B.C | HTA2, HTA1, HHT2 |
| 3 | A..EEE.AA...DD...ABCDE.AA...EE..A.A...CD.E..BD.B C.D..D....B.B..EEDCB...BBBC | HHF1, HTA1, HHF2, {HHT1, HTB1, HTA2} |
| 8 | AAE.E..AB..DDDB.BC...D.AA...EE...AB..CDD....C.E. A...DEE.AB..D...ED.B.BB | MCD1, YOX1, TOS4, {CLN2} |

Table 4.18 shows the longest genes with a support of 3. All of the histone patterns from the previous table have been compressed into 2 patterns, shown here as patterns 2 and 3 (and even these might be considered correlated enough to be a single pattern). The top ranked pattern here is formed from patterns 1 and 6 from the previous table. The last pattern has stronger biological significance than the similar patterns seen in Table 4.17; MCD1, YOX1 and CLN2 are all involved in mitosis with a $p$-value of 2e-5. Further YOX1, TOS4 [Horak, '02] and CLN2 are all SBF targets; [Flick, '98] however MCD1 is a target of MBF and not SBF [Simon, '01].

**Table 4.19. The longest patterns discovered with *K*=4**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | C.C....C.C...CB.CC.DDCB.BC.C..BD...C...DD..CB.B..C.EEB.B.BB....C.....C | RPL4A, RPS29B, RPP1A, RPP2B, {RPP0} |
| 2 | A..EEE.A....D....AB....AA...EE..A.A...C..E..BD.BC.D..D....B.B..EEDCB...B.BC | HHF1, HHT1, HTB1, HTA1, {HHF2} |
| 3 | D...C.DD..BBCC..D.....D.D.BB..B...D.....CC..BCDDD..B...D...D..BBB.CC..D | RSR1, POL2, DPB2, RAD51, {CDC45, MRC1} |
| 6 | D.B..D...B..C.C.C....AA...BC...DC.C.B.....B..DC....C.C.....E...DC..BC.CC.DD | NOP7, MRD1, YPL146C, YMR269W,{DIM1} |

Table 4.19 shows the first large group of entirely new genes. The first pattern consists of genes which are all involved in protein biosynthesis ($p$=1e-5). This is followed by a histone pattern ($p$=3e-13). The third pattern also consists of genes not seen in previous analysis. SGD shows a number of unique functional subgroups of these genes. POL2, DPB2 and CDC45 are all involved in DNA strand elongation ($p$=1e-6), whereas POL2, DPB2 and RAD51 are all DNA damage repair genes ($p$=0.0001) and RSR1, POL2, DPB2, RAD51 and CDC45 are all generally involved in cell proliferation ($p$=2e-5). Although SGD does not place MRC1 in a functional group at all, it is known to be an S-phase checkpoint protein found at replication forks, required for DNA replication and also required for Rad53p activation during DNA replication stress. This seems to correspond with functions of the other genes in this group. The final pattern here consists of 2 genes with unknown function, YPL146C and the now infamous YMR269W. This would likely confirm a role of YMR269W in rRNA processing. The other three genes in this group are involved in rRNA processing. This pattern agrees with patterns from the previous section (see Table 4.6).

**Table 4.20. The longest patterns discovered with *K*=5**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | C.DD..BBCC..D.....D.D..B..B...D.....C...B..D...B...D...D.. BBB.CC..D | RSR1, POL2, DPB2, RAD51CDC45, {MRC1} |
| 2 | C.C...CC.CCC....CC..D.B.B.B.B.B..DB....DDD | RPL4A, RPP0, HXT8, UTR1, EAF6 |
| 5 | C.....CB.CC.D.CB..C....B....C...DD..CB.B....EE..B.BB | RPL4A, RPS29B, RPP1A, RPP2B, RPS22A {RPP0} |
| 6 | ....D....A.....AA...EE..A.A...C.....BD.B..D..D....B.B....D.B ...B.BC | HTA2, HHF1, HHT1, HTB1, HTA1, HHF2 |

The first pattern here is virtually identical to pattern 3 in the previous table. The second pattern is difficult to confirm. It consists of two proteins already seen as being involved in protein biosynthesis. HXT8 is a hexose transporter, and it's inclusion in this gene group seems unfounded. The final 2 genes listed in this group have unknown functions. The next two patterns (not shown) are entirely supported by genes with unknown function (or dubious ORFs). The 5$^{th}$ ranked pattern is similar to the first pattern in Table 4.19 ($p$=1e-6). The last pattern, of course, contains histones ($p$=1e-15).

**Table 4.21. The longest patterns discovered with *K*=6**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | C.C...CC.CCC....CC..D.B.B.B.B....DB.....DD | RPL4A, UTR5, HXT8, EAF6, GAL2, RPP0, { YCL048W, YBL108W, YJL038C, HES1} |

The presence of a single pattern in this table is indicative of a strong deterioration in the quality of the patterns discovered. All 10 longest patterns, were all highly correlated and were combined to form the one pattern seen here. This shows a weak connection of genes, with a number of unknown genes and HES1 (a steroid synthesis gene). SGD shows that HXT8 and GAL2 are both carbohydrate transporters ($p$=0.0004). This $p$-value is only this low because so few genes are involved in carbohydrate transport. The genes RPL4A, RPP0 and HES1 are all involved, generally, in macromolecule synthesis ($p$=0.05) but there is nothing to justify the placement of UTR5 in this group nor to relate the functional subgroups together.

The above 5 tables would seem to indicate the pre-set value binning technique allows one to discover meaningful patterns in microarray data. However, the run times are prohibitively long for a desktop computer. The most notable new patterns were the protein biosynthesis pattern (see Table 4.19) and the DNA strand elongation and DNA damage repair pattern (see also Table 4.19). In addition to these novel patterns the ubiquitous histone grouping is evident in many of these tables (particularly Table 4.20) as is an rRNA processing group (Table 4.19).

#### 4.1.4.5 K-means Clustering Technique

This technique uses the Xmeans implementation [Pelleg, '03a] of K-means clustering to choose the best value of $K$ for each gene. Doing this allows one to take the continuously valued microarray data and discretize it into $K$ bins on a gene-by-gene basis. After executing Xmeans, the centroids will be merged into 3 bins (see Section 3.2.5).

Merging the centroids had the side-effect of causing many patterns to form. Pattern discovery was unable to finish with parameters $L$=4, $W$=9 and $K$=2 (support). Even employing the Beowulf clustered required $L$ to be increased to 6 before pattern discovery could finish in 24 hours, at 3252 seconds. The parameters $L$=5, $W$=9 and $K$=6 were also tried but that could not finish in under 24 hours. Using higher values of $K$ was not attempted because lower values had already produced very good results with the other two binning techniques. Lower values of $W$ were not used because it places stronger

restrictions on the density of the patterns discovered. Increasing both *L* and *W* was not attempted.

The patterns discovered are presented below.

**Table 4.22. Longest patterns of support 2**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | MAA...AAAAAAA.AAAAAAAAAAAAAAAA.AAAAAAAA.AAA.AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA | GAL3, GAL2 |
| 2 | ZZ...AAMMMMMMMMMMMMMMMAAMMMMMMMMAAM.MMMMMMMM..MA.MMMMMMMMMMM..MMAAAAMMMM.MMMMM | TUB1, SIM1 |
| 3 | ZZAAAAAAMZMAAAAAMZZ.AAAAZZ.AAAAAA..Z.MAAAAAAAAAAAA.AAAAAA.AA.AAAAAAAAMZ..AAAA | HTA2, HHT2 |
| 4 | MA.AAA..AAAAAAA.AAAAAAA.MAAAAAA.MZA.AAAAAAA.AAAA.AAAAAAAZM.AAAAAAAAAAAAAAAAA | KCC4, TOS2 |

Because of the lack of diversity in the template of these patterns (the preponderance of a single character) one might expect these patterns to simply be noise. However, due to their sheer density these patterns do convey some biological meaning. The first pattern is supported by two galactose genes ($p$=9e-7). In the second pattern, both genes deal with microtubule cytoskeleton organization and biogenesis ($p$=0.001). The third pattern contains histones ($p$=1e-5) and the last patterns contains 2 genes that have little functional annotation but the gene-products co-locate to the bud neck ($p$=0.0002).

**Table 4.23. Longest patterns of support 3**

| Rank | Pattern | Supporting Genes |
|---|---|---|
| 1 | AA...AAAAAAA.AAAAAA..AAAAAAA..AAAAAAA.AAA.A.AAAAAAA..AAAAAAAAAAAAAAAAAA | GAL3, GAL2, TOS1 |
| 2 | A..AAAAAAAAAAAAAAAA..AAAAAAA..AAAAAAAAAA..AA.AAAAAAA..A.AAA.AAAAAAAAAA... | GAL2, GSY2, TOS2 |
| 3 | AM.M..MMMM.MMM.MMMMMM..MMMM.MM.MMMMMMMMMM.MMM.MMM.MMMMMMMM.AMM.MMM..MM | DIA3, PIG2, IFH1 |

The first pattern here is the galactose pattern from above with the addition of TOS1 whose function is unknown. The second pattern contains GYS2, another gene involved in carbohydrate metabolism along with GAL2 ($p$=0.00246) and TOS2 which functions in the bud-neck. In the last pattern, DIA3 and IFH1 are both involved in the general theme of development ($p$=0.01), whereas PIG1 is a protein phosphatase regulator. Interestingly, PIG1 regulates GSY2.

For all values of $K > 3$ the top patterns discovered have very little biological significance as could be determined by their GO annotations. This somewhat disappointing result points to the unrefined nature of this novel microarray binning technique. Because this technique involves 2 steps to bin the data (Xmeans clustering and merging of those bins into fewer bins) there is more room for error in the technique and more room for improvement. The first possibility for improvement would be to use more bins as input to Teiresias. 5 or 7 bins would speed up pattern discovery and may also make the patterns found more specific, which in turn could increase their biological significance.

### 4.1.5 Hierarchical Clustering with Cluster

In this section, it is examined whether the biologically most relevant patterns from the each of the previous 3 sections are found grouped together by the hierarchical clustering

method of Cluster (which uses the Pearson's correlation coefficient as a distance metric, see Section 3.2.8). In order to do this Cluster was executed on the same 612 genes as used above. In some cases multiple patterns are indicative of the same biological relationship. For example, many patterns contained histone genes. In such cases, the largest (most genes) representative pattern is selected to be studied. In order, the patterns to be examined are:

1) The rRNA processing group (from Table 4.6): NOP7, HCA4, MRT4, EBP2, MRD1, DIM1, YMR269W, YPL146C, NOP6.
2) The bud-neck group (from Table 4.8): MOB1, ACE2, CDC5, HOF1, YIL158W, YML119W.
3) The cell proliferation group (from Table 4.9): RFA1, KCC4, MCD1, MSH6, RSR1, SPT21, ASF1, YCL022C
4) The least probable pattern found by Genes@Work (from Table 4.10): FIG1, PHO5, HXK1, PNC1, SPO12, RAD27, RMA1, SUN4, CHS1, YCRX18C, YER067W, YHR087W.
5) The histone group (from Table 4.15): HHT1, HTB1, HTA1, HHF2, HHT2, HHO1 HTA2, SVS1.
6) The DNA-replication group (from Table 4.15): POL30, KCC4, MCD1, MSH6, MSH2, YCL022C.
7) The mitosis/SBF group (from Table 4.18): MCD1, YOX1, CLN2, TOS4.
8) The DNA-damage/elongation group (from Table 4.19): RSR1, POL2, DPB2, RAD51, CDC45, MRC1.
9) The protein biosynthesis group (from Table 4.18): RPL4A, RPS29B, RPP1A, RPP2B, RPS22A, RPP0.
10) The galactose group (from Table 4.23): GAL2, GAL3, TOS1.

For each pattern it will be determined whether a similar group is formed in the results from Cluster. The results from Cluster will be considered better if the cluster in Cluster includes more genes that are related to the core function of the pattern's supporting genes (for example, including more histone genes in a histone cluster), or if it excludes genes that are not easily identified as being biologically related to the core function of the

group (such as excluding SVS1 from the histone group). Cluster's results will be considered worse if it excludes genes from the pattern that are central to the pattern's function (such as removing NOP6 from the rRNA-processing group).

Because all genes are clustered together at some point in a hierarchical tree, the correlation (Pearson's *r*) value for the smallest cluster that contains all the genes in the pattern will be listed. Also, the number of genes that did not appear in the pattern but did appear in the Cluster grouping will be listed. In addition, the number of genes that are correlated in the entire data set at this *r*-value or lower, as determined by a short script, will also be shown. This was done because the *r*-value is the only score provided by Cluster to determine the "goodness" of a cluster. Although an *r*-value of 0.80 might appear significant and indeed, is statistically significant, it is important to realize that many genes are clustered together at that *r*-level and higher. For example, at an *r*-value of 0.92, only 24 genes are involved in a relationship. Thus any gene clustered at a value of 0.92 is in the top 30 of most significant clusters in this dataset. At an *r*-value of 0.80, approximately 180 genes or 1/3 of all genes are involved in a cluster. Therefore, there may be nothing to mark that cluster as significant to a researcher, and differentiate it from any of the other sub-trees formed by Cluster, whereas the patterns presented here are the most interesting of all the patterns discovered.

The first 8 of 9 genes in the rRNA processing group are clustered with a an *r*-value of 0.78 with one other gene (AAH1, a purine salvage gene) by Cluster. However, to include NOP6 in this group requires the *r*-value to be lowered to 0.57. For the smaller group, ~230 genes are clustered together at the same or higher *r*-value, whereas over 500 genes are clustered together with an *r*-value of at least 0.57. From this data it would appear that pattern discovery does a better job at finding this functional group than does Cluster. Although AAH1 has some biological significance to any set of genes involved in DNA metabolism or catabolism, NOP6 is more closely related to the core function of this set of genes.

The bud neck cluster is found in the Cluster results as a single group with an *r*-value of 0.76 but is found with 10 other genes. Only two of the 10 extra genes – MMO1 & MYR1 – also co-localize to the bud neck, whereas the majority of the extra genes are involved in meiotic cell-cycle. At such an *r*-value approximately 250 genes are involved in a group of some kind in this data. In this case, it would appear that pattern discovery does a better job of picking out this bud neck group then does Cluster. Indeed, examining the results from Cluster with respect to their gene ontology data would lead to the conclusion that this group was primarily involved in the meiotic cell-cycle (a very general term in which many genes fit), and had little to do with the bud neck region of the cell. The merging of these distinct functions likely occurs because the expression values of the bud neck genes and the meiotic-genes are "close enough" to be placed in a single cluster. For this case pattern discovery is more robust to this noise, and pulls out "bud neck" genes as being more related. Further, unlike most of the gene clusters discussed here, the genes in the bud neck pattern do not share a common GO annotation in terms of biological process. This might put some doubt onto the validity of this cluster but, the fact that 2 different analysis techniques would cluster these genes together (to varying degrees) would seem to confirm that they are co-regulated in the cell, combined with the knowledge that the products from these genes co-localize at the bud neck seems to confirm the biological relevance of the cluster and might imply a more tightly knit biological function than is currently understood for these genes.

For the cell proliferation group, Cluster groups RFA1, MCD1, MSH6 and RSR1 together (missing 4 genes: KCC4, SPT21, ASF1, YCL022C) at an *r*-value of 0.85. This cluster also includes 5 other genes. Six of the genes 9 grouped by Cluster are involved in DNA-replication. A large cluster which includes all of the genes from the pattern is found at an *r*-value of 0.57. However, these genes are not well characterized as having a single, unifying biological role. Because this pattern's biological role was the general "cell proliferation" – it would seem that the smaller group discovered by Cluster is separating out the DNA-replication genes from the other genes listed in the pattern. This result may indicate that the gene group discovered by Teiresias has a core function of DNA-replication, the other genes that are clustered with these genes that are not involved in

DNA-replication could therefore be considered noise. Under such a hypothesis, Cluster's results are better because they highlight the DNA-replication role of these genes and reduce noise.

The pattern found in Table 4.10 was chosen because it was the pattern determined to be least probable by Genes@Work in Section 4.1.4.3. The genes in this cluster at not clustered together in Cluster until an *r*-value of 0.0027. In this cluster are fully 2/3 of all the genes in the data set. Because this pattern has very little biological significance, the fact that Cluster does not produce it is a point in favor of Cluster. However, it should be noted that this pattern is a real occurrence in the data and not the result of some mistake in the algorithm. The reason that this pattern is found at all is because of pattern discovery's ability to find very subtle similarities (short patterns) because they are supported in many streams. If such a small number of similarities were important to forming a biological relationship we might see better results from Genes@Work's scoring system. However, it appears that subtle similarity is not an important factor for this data. That is, most of the biologically significant clusters as confirmable by their GO annotation will occur with strong correlations.

The histone pattern, one of the clearest and most easy to confirm patterns, is also found by Cluster. Cluster, however, manages to include 2 more histones in the grouping at an *r*-value of 0.81 (where approximately 180 genes have been clustered together in some way). However, the cluster does not include SVS1. With the biological information available, it would appear that Cluster does a better job of forming a biologically cohesive group than pattern discovery. However, if SVS1 is implicated in having an effect on histones then it might be concluded that pattern discovery finds a different, but equally significant cluster.

The DNA-replication group is partially found in Cluster. All the genes except YCL022C and KCC4 are found together in a small group of 7 genes with a correlation value of 0.88 (at which point only ~50 genes are clustered). YCL022C and KCC4 are found together with a correlation of 0.92 (only 24 genes have been clustered at that point). Both clusters

are merged at an *r*-value of 0.69 in a group that contains 41 genes. The role of KCC4 in this group is questionable thus it's exclusion by Cluster would favor the Cluster results.

The mitosis/SBF group is found in Cluster with a reasonable correlation of 0.78 along with 15 other genes. In this super-set of genes, however, the mitosis relationship of these few genes is lost because many (7) of the genes in this group are DNA damage repair genes. Similarly to the bud neck group above, smaller functionally related groups of genes are merged into a single, larger group and the relationship is lost. By allowing genes to be part of many groups, pattern discovery allows the detection of more subtle groups of genes, as evidenced here. In this respect, pattern discovery is superior to hierarchical clustering.

The DNA-repair and elongation group is also found in the results from Cluster. All the genes except CDC45 are found in a single group with 14 other genes at an *r*-value of 0.79. Of these 14 extra genes, only 2 are also involved in DNA-repair and only 1 is involved in DNA-strand elongation. Thus, the Cluster results dilutes the DNA-elongation characteristic of this group. CDC45 is included in the cluster at an *r*-value of 0.75 along with 25 other genes. Of these 31 total genes, 12 are involved with DNA repair and only two new genes are involved with DNA-elongation. Unlike some other clusters above, where an outlier gene (KCC4) has a questionable relationship to the function of the cluster, CDC45 is very functionally related to this group. Therefore, it would seem that pattern discovery forms a better cluster than Cluster for these genes with respect to their DNA-elongation biological function.

The protein biogenesis group occurs in Cluster at an *r*-value of 0.62 along with 5 other genes. Four of these 5 genes are also involved in protein biosynthesis. At this level of r over 420 genes also occur in a cluster of some kind. This cluster is interesting because it is a very functionally related group, occurs at a low *r*-value, yet contains very few genes. Because of this, such a cluster would be over-looked by a researcher as it implies that the genes aren't clustered together because they are similar, but excluded from other groups because of dissimilarity. Yet, because these 6 genes are co-regulated tightly at a few

points in time, they form a strong pattern. This again highlights the importance of balancing the pattern length with its support.

The galactose group from Table 4.23 is not found with TOS1 in Cluster. However, GAL2 and GAL3 are found together with GAL10 at an *r*-value of 0.85. The clustering of TOS1 in this group is not obvious, therefore it seems that Cluster has done a better job of forming a biologically cohesive group.

In this section the yeast cell cycle data was used to benchmark Teiresias, Splash and Genes@Work, as well as to evaluate 3 different binning techniques and for the sign-of-the-derivative technique to compare 2 orderings of the patterns; the Genes@Work ordering and ordering by pattern length. Finally, the most biologically significant patterns were compared to the results from Cluster.

It was shown that with this data set, Teiresias is over 5 times faster than Splash on a data set, with parameters that produce a large number of patterns. However, Splash is considerably more memory efficient than Teiresias on any size result set. Genes@Work was considerably slower, and less memory efficient than both Teiresias and Splash. This is because Genes@Work uses the java programming language which is generally slower than C, and less memory efficient as well. Further, Genes@Work implements a GUI, and a number of other data analysis features over and above pattern discovery. This adds to Genes@Work's memory foot-print. However, even taking this into account, Genes@Work is less memory efficient.

In terms of utility, the sign-of-the-derivative method was the best of the three tested binning methods; followed by pre-set value and Xmeans. Sign-of-the-derivative found many biologically significant patterns, was easy to implement and finished much more quickly than the other two methods. It is likely that the sign-of-the-derivative technique found the best patterns because the biologically significant relationships are best captured by the '+/-/0' mapping. Although both pre-set value, and Xmeans methods found some patterns that were not found by sign-of-the derivative, the results were not significant

enough to make them better choices. The Xmeans methods preformed very poorly, in both execution times, and in terms of biologically significant patterns found. Further use of this method would require a great deal of refinement.

For two cases, Genes@Work's ordering method produced biologically significant patterns. However, in many instances it did not. Generally, biologically significant patterns were more highly concentrated in the longest 5 or 10 patterns for any particular support level. Although the training data given to Genes@Work was artificial, the application still should have been able to balance support and pattern length in a meaningful way. Thus for this type of pattern finding problem, Genes@Work's ordering is not better than ordering by the longest pattern.

Generally, Cluster found many of the same functionally related gene groups as pattern discovery. In many cases it found superior groups, clustering together other genes that shared the same function as the pattern's genes. Cluster also executes much more quickly than pattern discovery, usually in less than 60 seconds, and is much easier to use than Teiresias, Splash or Genes@Work. However, hierarchical clustering is inferior to pattern discovery in two ways. First, specific gene relationships are lost in large groups; a pattern may find a DNA-elongation group, but Cluster will group these genes in a larger group who's unifying functional relationship is a very broad or general term, such as "cell proliferation" or "cell cycle". Often, a researcher is interested in a gene group's function at a finer level. Second, hierarchical clustering only allows one gene to be part of one group. Thus a gene that has multiple functions, and is therefore likely regulated by multiple factors, is forced to be part of one group or another. In the worst case, it would be possible for the gene to be placed into a "compromise group" that was not involved with either function. In such a case, the finer relationships and function of a gene may be lost.

## *4.2 Lymphoma Results*

This section uses the data of Alizadeh [Alizadeh, '00]. The original analysis of this data used hierarchical clustering with Cluster to form 2 subtypes of diffuse large B-cell

lymphoma (DLBCL): Germinal Centre B-cell (GCB) and Activated B-cell (ABC). This was done using 47 patient samples with 380 genes. This analysis is essentially a classification of DLBCL by clustering and will be referred to as such. Throughout this section the patient samples will be referred to by the order in which they were placed by Cluster. The first 24 samples (0-23) are of the GCB type and the next 23 samples (24-46) are of the ABC type. Because the work by Alizadeh is well respected and highly cited, with many papers building on the original work, it will be used as the benchmark to measure the usefulness of pattern discovery to cluster this data. In Figure 4.4, we see the original clustering of DLBCL subtype, with added numbering of the tissues.

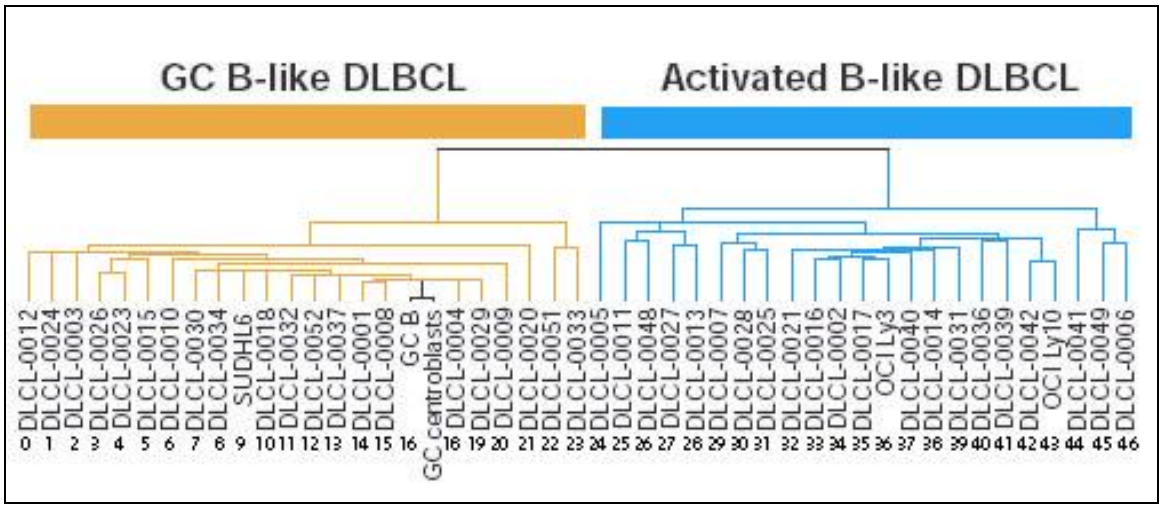


**Figure 4.4. Hierarchical clustering of DLBCL subtypes by Alizadeh, with added numbering of tissue subtypes.**

Section 4.2.1 examines how well pattern discovery, using Teiresias, can cluster this data using the Xmeans stratification technique. Evaluating the quality of the patterns is a 4-step process. First, general properties, such as maximum length and support, of the patterns are determined. Second, a list of the highest support patterns that agree perfectly with the results from Alizadeh is constructed. From this list, the number of genes occur in each pattern, and the support level of the top pattern is noted. Third, using the data from the first and second step, a series of orderings (rankings) with minimum support or length requirements is made. These patterns are then examined to see how well they

agree with the results from Alizadeh. Four, the patterns from the ordering scheme that yielded the most discriminating top 20 patterns are selected and used to compose a list of genes (candidate genes) that are believed to discriminate between the two subtypes. This sections ends with a brief examination of a potential method to automatically cluster patterns based on their correlation and an investigation into cross-clustering tissues.

Section 4.2.2 follows the same methodology as Section 4.2.1 and the candidate genes from both sections are compared. The data for Sections 4.2.1 and 4.2.2 are available in Appendix A. In order to save space, Appendix A does not contain all the data generated for these experiments, but does contain the data most pertinent to the results. In Section 4.2.3 the Genes@Work approximation experiment will be discussed briefly.

Section 4.2.4 the results from Genes@Work will be examined. The levels of support for the highest supported patterns will be compared to those found by Teiresias. Also, the quality of Genes@Work's own ordering scheme will be investigated. Lastly, the highest support patterns found by Genes@Work will be used to form a list of genes that are believed to discriminate between the two subtypes of DLBCL.

### 4.2.1 Clustering with Xmeans

Xmeans bins each gene into $K$ different bins (centroids). The value for $K$ can be different for each gene. This makes this binning technique gene-specific, rather than global, such as the pre-set value technique. For this data set, running Teiresias with the parameters $Z$=1, $K$=2, $L$=2, $W$=3 all patterns are discovered on the Beowulf cluster in <24 hours.

Because so many characters are used as break-characters there is a need to limit t the maximum number of bins used in Xmeans by some value, $K_{max}$. Four upper limits were tried: $K_{max}$=5, 10, 15 and 20. However, 5 was not used for further analysis because with this value over 340 genes are mapped to a single bin[14] and the remaining 40 genes were considered too few to give meaningful results. Therefore, 3 data sets were executed on the Beowulf cluster.

[14] Genes that have just one bin are non-discriminatory and therefore are removed from analysis.

The execution times and number of genes used are given in Table 4.24. The execution times are influenced primarily by 2 things. As $K_{max}$ increases the number of characters used also increases, which reduces the number of patterns found, generally, and therefore execution time also decreases. However, as $K_{max}$ increases, more and more genes are included in the data set which will increase execution time. This is why a decrease is seen in the execution times between $K_{max}$=10 and $K_{max}$ =15, but an increase with $K_{max}$=20.

**Table 4.24. For each data set the number of genes that had more than one bin, and the execution time on the Beowulf cluster is given**

| $K_{max}$ | Number of Genes Used | Execution Time (s) |
|---|---|---|
| 5 | 33 | Not Executed |
| 10 | 186 | 5994 |
| 15 | 294 | 4141 |
| 20 | 358 | 10479 |

After pattern discovery was completed, each result set was analyzed in a 4-step process. First, for each result set, several orderings of the patterns were made. The patterns were ordered by support and then by the length of the pattern. The first ranking will show which genes are common to many of the tissues. The second shows the most closely related tissues and gives an upper limit on the expected number of common genes that would be shared by any cluster. Then a supervised ordering of the patterns was done where patterns were removed if not 100% supported by ABC or GCB tissue types, and the remainder were ordered by support. This ranking gives an upper bound on the maximum support/genes a perfectly discriminatory pattern would have. Using these upper bounds, two more complex types of orderings were carried out. The first required the patterns to have a certain level of support, then they were ordered by length. The second required the patterns to have a certain minimum length, and then were ordered by support. This was done to see if there was a point where all of the top 20 patterns were supported only by ABC samples or GCB samples. If this did occur the ordering could be used as a general guide to picking out patterns that could be used as classifiers.

Pattern discovery on all three data sets produced a number of genes with very high levels of support. The range of support for the top 20 patterns for each data set was 23-35, 22-34 and 21-34 for $K_{max}$=10, 15 and 20, respectively (Table A.1 & Table A.2). As might be expected, most of the patterns from all 3 data sets were poor classifiers of DLBCL subtypes. None of the supporting tissues were ever compose of more than 87% of either class, and the majority were less than 70%. Although these patterns are not likely to be useful as classifiers, these patterns have value in indicating which genes would be poor classifiers of DLBCL subtypes. That is, given a list of candidate genes that could be used to classify subtypes of DLBCL, we would expect some of them to be genes found to have very high support across all tissue types. These genes would then be removed from the candidate list because they have poor discrimination.

Next the patterns were ordered by length (Table A.3 & Table A.4). The top 20 patterns – in terms of the supported tissues – for all three data sets were very similar, although the ordering was often different. There are four important notes to make about this data. One, not unexpectedly, the top patterns were supported by only 2 tissues. Two, the longest patterns contained 76, 95 and 99 genes, for $K_{max}$=10, 15 and 20, respectively. This translates into 40.9%, 32.3% and 26.8% of the maximum possible number of genes. Three, the majority (70%) of the top patterns were supported by ABC tissues. This would seem to indicate that the ABC subtype is more tightly knit in terms of genetic expression than GCB when using this binning technique. Four, it should be expected that all the top 20 longest patterns would have perfect discriminatory power between GCB and ABC tissue types because a long pattern means that a large number of genes express similarly across that two or more tissues. However, for all three data sets this was not the case; there were always 3 patterns (15%) that consisted of 2 tissues, one from each subgroup. These same three patterns were found in each of the three data sets and identifies the following pairs of tissues as cross-clustering: {4,24}, {21, 33} and {21, 41}. These results will be discussed further below. However, it should be noted that, in data that has been grouped by Cluster, tissues that are found at the boundary of a cluster are less correlated with the cluster than tissues that are found at the centre of the cluster.

Thus, the tissues 0 and 23 (boundary tissues to the GCB subtype) are less correlated to the GCB cluster than are tissues 12 and 13 (tissues that are central to the GCB subtype). Likewise, tissues 24 and 47 (boundary tissues to the ABC subtype) are less correlated with the ABC cluster than are tissues 35 and 36, which are central. Thus it is more likely that boundary tissues could have been misclassified by Cluster. Graphically this can be seen in Figure 4.4. Observe how samples 21-23 and 0-2 join the GCB cluster later (higher) on the dendrogram, whereas samples 14-19 form the center of the cluster. Likewise, tissues 44-46 and 24 are outliers to the ABC group.

The second step of analysis is to find the largest level of support that showed perfect discrimination between ABC and GCB subtypes, all patterns that were not 100% supported by either group were eliminated and the remaining patterns were ordered by support. The top 20 patterns for each data set was examined and a summary of the data is in Table 4.25. See Table A.5 & Table A.6 for more detailed results.

**Table 4.25. The percentage of patterns of supported only by the ABC subtype, the maximum support for ABC subtype patterns, and the maximum support for GCB subtype patterns from the top 20 highest support patterns**

| $K_{max}$ | % Patterns ABC | Max. support for ABC | Max. Support for GCB |
|---|---|---|---|
| 10 | 80% | 18 | 18 |
| 15 | 75% | 17 | 18 |
| 20 | 100% | 16 | - |

As can be seen in Table 4.25 all three data sets favor patterns involving the ABC subtype. This would imply that the ABC subtype is more "tightly knit" than the GCB type using this binning technique. However, it can be seen, because the maximum support for a GCB pattern is equal to or larger than the maximum support for an ABC pattern, that a single pattern can capture an equal or larger portion of the GCB subtype than can any single pattern for the ABC subtype. In most cases the pattern consists of 2 or 3 genes, although patterns with 1, 4 and 5 genes are also present. This data also gives some

credence to being able to classify subtypes of DLBCL using pattern discovery. It shows that patterns with reasonable numbers of genes can account for a large portion (~80%) of a particular subtype.

In step 3 of the analysis, the data from steps 1 and 2 is used as a guide to forming more complex orderings of the patterns. These orderings involved selecting patterns that met some minimum criteria in terms of length or support, then ordering by support or length, respectively. The minimum values for length were $G_{min}$ = 2, 3, 4 and 8 genes and the minimum values for support were $S_{min}$=5, 10, 14 and 16. The values for $G_{min}$ were chosen because the highest support patterns with perfect discrimination had between 2 and 5 genes. So, genes in this range, or slightly above, should give patterns with good discrimination. Values for $S_{min}$, were chosen to reasonably cover support values less than or equal to 16, because the smallest maximum support for a pattern with perfect discrimination is 16 (see Table 4.25). In all cases – except $S_{min}$=16, $K_{max}$=20, where two patterns were ~85% supported by GCB tissue samples – the top 20 patterns were majority supported by ABC type DLBCL, and not GCB. Because of this, the ABC clustering patterns will be discussed here, which will be followed by an examination of the GCB clustering patterns below. In Table 4.26, we can see a summary of the ABC-pattern orderings. In Appendix A, more complete data is presented for $K_{max}$=10: $G_{min}$=8 (Table A.7), $S_{min}$=5 (Table A.11), $S_{min}$=10 (Table A.15) and for $K_{max}$=20: $G_{min}$=8 (Table A.8), $S_{min}$=5 (Table A.12), $S_{min}$=10 (Table A.16).

**Table 4.26. For each data set ($K_{max}$=10, 15, 20) the number of top 20 patterns whose supporting tissues consist of 80%, 90% or 100% ABC subtype for each type of minimum gene criteria ($G_{min}$=2, 3, 4, 8) or minimum support criteria ($S_{min}$=5, 10, 14, 16)**

| Criteria | $K_{max}$= 10 | | | $K_{max}$=15 | | | $K_{max}$=20 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 80% | 90% | 100% | 80% | 90% | 100% | 80% | 90% | 100% |
| $G_{min}$=2 | 11 | 3 | 0 | 12 | 4 | 0 | 8 | 2 | 0 |
| $G_{min}$=3 | 19 | 7 | 1 | 19 | 9 | 3 | 19 | 9 | 3 |
| $G_{min}$=4 | 20 | 12 | 4 | 20 | 12 | 4 | 19 | 15 | 7 |
| $G_{min}$=8 | 17 | 6 | 6 | 20 | 0 | 14 | 19 | 18 | 18 |
| $S_{min}$=5 | 19 | 19 | 19 | 18 | 18 | 18 | 17 | 17 | 17 |
| $S_{min}$=10 | 20 | 16 | 7 | 20 | 17 | 6 | 20 | 19 | 10 |
| $S_{min}$=14 | 20 | 12 | 3 | 20 | 12 | 4 | 17 | 12 | 3 |
| $S_{min}$=16 | 17 | 7 | 2 | 18 | 8 | 2 | 11 | 3 | 0 |

There are three important factors that concern the data in Table 4.26. First, the number of good patterns increases with increasing $G_{min}$. The only exception to this might be for $G_{min}$=8, $K_{max}$ = 10, where there may be some deterioration in the ability of the patterns to discriminate between subtypes. However, even in that case the number of 100% discriminating patterns increases. The quality of the patterns improves because, as $G_{min}$ increases, the tissues clustered together must be more related. Second, the discrimination ability of the patterns increases with decreasing $S_{min}$. At $S_{min}$=5, for $K_{max}$ = 10, 95% of the patterns are 100% discriminating. This occurs because, as the minimum required support decreases, the number of genes in the pattern increases – 16, 8, 5 and 4 genes for $S_{min}$=5, 10, 14, and 16, respectively. Third, the effect of a low $K_{max}$ is favorable with a low $S_{min}$, but unfavorable with a high $G_{min}$. The reason, generally, why a higher $K_{max}$ does a better job of discriminating the subtypes for increasing $G_{min}$ is because the support for patterns is less in $K_{max}$=20 than for $K_{max}$=10. For instance the highest support pattern for $G_{min}$=8, $K_{max}$ =10 and $K_{max}$ =20 is 10 and 8, respectively. The reason why $K_{max}$ = 10 out performs $K_{max}$ =20 with decreasing minimum support is because the patterns with $K_{max}$ = 10 generally have more genes in them than in $K_{max}$ = 20. For instance, the shortest top 20 pattern for $S_{min}$=5, $K_{max}$=10 and $K_{max}$ =20 is 15 and 13 respectively.

Having examined the strongly discriminating patterns in the above table, it is also of interest to examine which tissues are misclassified by these patterns. Looking at $S_{min}$=5, $K_{max}$=10, there is only one poor pattern. This pattern clusters together the following tissues {11, 21, 24, 33, 38}. The situation is similar in $G_{min}$=8, $K_{max}$=20: Here, two patterns fail to discriminate at 100% efficiency. Their supporting tissues are as follows: {9, 13, 35, 37, 40, 41, 44} and { 22, 28, 34, 37, 39, 41, 42, 43}. From the data presented at the beginning of this section, it is perhaps not too surprising to see tissues 21 and 24 mis-classifying. Tissue 22 is also an outlier to the GCB cluster and therefore would be more expected to cluster with ABC tissue. However, the fact that tissues 9, 11 and 13 would misclassify seems perplexing as these should be central to the GCB cluster.

Because the ABC subtype dominated the above orderings, there is no information on the quality of the GCB patterns. However, in a somewhat artificial approach, it is possible to remove all patterns that aren't at least 50% GCB and examine how well those patterns classify the GCB subgroup. This roughly simulates an algorithm that has already clustered together the stronger ABC subtype and is left with predominately GCB supported patterns. Another way to regard this technique is to view patterns that are supported by >50% ABC tissues as patterns that "cluster ABC tissues and potentially cross-cluster GCB tissues", and patterns that are supported by >50% GCB tissues as "clustering GCB tissues and potentially cross-clustering ABC tissues". Then this technique simply removes the patterns that cluster together ABC tissue samples and leaves the GCB patterns. Because the results in Table 4.26 indicate that $K_{max}$=10 and $K_{max}$=20 yield the best results, we focus on only those two data sets in Table 4.27. In Appendix A more complete data is presented for $K_{max}$=10: $G_{min}$=8 (Table A.9), $S_{min}$=5 (Table A.13), $S_{min}$=10 (Table A.17) and for $K_{max}$=20: $G_{min}$=8 (Table A.10), $S_{min}$=5 (Table A.14), $S_{min}$=10 (Table A.18).

**Table 4.27. For each data set ($K_{max}$=10, 20) the number of top 20 patterns whose supporting tissues consists of 80%, 90% or 100% GCB subtype for each type of minimum gene ($G_{min}$=2, 3, 4, 8) or minimum support ($S_{min}$=5, 10, 14, 16) criteria**

| Criteria | $K_{max}$= 10 | | | $K_{max}$=20 | | |
|---|---|---|---|---|---|---|
| | 80% | 90% | 100% | 80% | 90% | 100% |
| $G_{min}$=2 | 4 | 3 | 2 | 3 | 2 | 0 |
| $G_{min}$=3 | 8 | 7 | 3 | 9 | 4 | 0 |
| $G_{min}$=4 | 15 | 13 | 10 | 12 | 9 | 3 |
| $G_{min}$=8 | 14 | 14 | 14 | 10 | 5 | 5 |
| $S_{min}$=5 | 5 | 3 | 3 | 5 | 2 | 2 |
| $S_{min}$=10 | 15 | 13 | 10 | 12 | 8 | 2 |
| $S_{min}$=14 | 4 | 3 | 2 | 4 | 0 | 0 |
| $S_{min}$=16 | 4 | 3 | 2 | 3 | 1 | 0 |

Generally, the quality of the GCB clustering patterns is less than it was for the ABC patterns; there are fewer patterns at every level of discrimination. This is likely because the GCB subgroup is less well-defined than the ABC subgroup. That is, its genes are not as "tightly" binned as those in the ABC subgroup. It is not unexpected then, that $K_{max}$=20 does a worse job on this data then $K_{max}$=10. It would seem that $K_{max}$=20 is simply too fined-gained for the variability encountered in the GCB tissue samples. Another dissimilarity between the GCB and ABC results is that $S_{min}$=5 provided much worse results than $S_{min}$=10. This occurs because 4 tissue samples (24, 29, 31 and 33) support many of the top 20 patterns and this cross-clustering reduces the discriminating power of the patterns. Although these same tissue cross-cluster to some extent when $S_{min}$=10 (along with tissues 44-46), the effect of this is diluted because of the higher minimum required support. Looking to the tissues that were mis-classified in $G_{min}$=8, $K_{max}$=10, we find that tissues 24, 29, 31 and 38 are misclassified multiple times. Tissue 24 has repeatedly shown problems with cross-clustering with GCB. However, tissues 29, 31 and 38 should be central to the ABC cluster and should not misclassify.

From a clinical standpoint, the end-goal of any microarray classification analysis such as this is to produce a set of genes that can be indicative of the subtype a patient might

belong to. Classifying patients in this manner can determine the treatment that they receive. Clinically, it is important to identify the smallest set of genes that can classify a subgroup because a test that requires more genes to be effective is more costly. In the fourth step of analysis, a list of genes is presented here that seem to do a good job of identifying ABC or GCB subtype.

The genes used to classify the ABC subtype are the amalgamation of all genes in the top 20 patterns from $K_{max}$=20, $G_{min}$=8 (see Table A.8). Using this set of patterns keeps the total number of genes small. It also reduces the number of cross-clustering tissues (false positives) but at the cost of reducing the number of ABC tissues included (true positives). Doing this clusters together the following tissue samples: 28, 30, 32, 34, 35, 36, 37, 39, 40, 41, 42, 43, 44 and 45; or 14/23 ABC tissues. It also cross-clusters the following GCB subtype tissues: 9, 13 and 22 (3/24). This clustering is achieved with the following genes:

- FMR2 (Fragile X mental retardation 2. LAF4 and AF4 homologue)
- MYBL1 (myb related gene A)
- DCTD (Deoxycytidylate deaminase)
- KIAA1039
- CD10
- ALOX (Arachidonate-5-lipoxygenase)
- PI3K (Phosphatidylinositol-3-kinase)
- BCL2
- CD21 (B-lymphocyte CR2 receptor)
- TDT (Terminal Deoxynucleotide Transferase)
- KIAA0870
- ETV6 (TEL oncogene)
- Hs.192708 (highly sim. to mybA)
- Hs.106771
- Hs.125815
- Hs.24724
- Hs.203866
- Hs.124049
- Hs.97530
- Hs.28355
- Hs.75859
- Hs.224323
- Clone.1234554

- Clone.825920
- Clone.1288046
- Clone.1334486

The genes used to classify the GCB subtype are the amalgamation of all genes in the top 20 patterns from $K_{max}$=10, $G_{min}$=8, ABC<50% (see Table A.9). In a similar vein as above, this keeps the number of genes used down to a minimum, and also keeps the cross-clustering rate low. Doing this clusters together the following GCB tissues: 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 18, 19, 21, and 23; or 17/24 GCB tissues. It also cross-clusters the following ABC tissues: 24, 29, 31, 33, 38 and 39 (6/23). The genes identified to do this are as follows:

- KIAA1039
- MYBL1 (MYB related gene A)
- DCTD (Deoxycytidylate deaminase
- TRKC (Neurotrophic tyrosine kinase receptor type 3)
- ID2H (Inhibitor of DNA binding 2)
- PTK (NET PTK tyrosine kinase)
- RAD50
- CAM1 (CAM-kinase I)
- MYOIC (Myosin IC)
- ALOX (Arachidonate-5-lipoxygenase)
- CD27
- BRCA2
- JMJD1B (Putative zinc finger protein)
- WASPIP (HS-PRPL-2-WASP interacting protein)
- OGG1 (8-oxoguanine DNA glycosylase)
- ALU subfamily J
- FAK (Focal adhesion kinase)
- CD10
- RGS13 (Regulator of G-protein signaling)
- TCEB3 (Elongin A SIII p110 subunit)
- ALU subfamily C
- TDT (Terminal Deoxynucleotide Transferase)
- CSNK1G2 (Casein kinase I)
- Hs.28355
- Hs.120716
- Hs.203866
- Hs.75859

- Hs.222808
- Hs.19399
- Hs.161905
- Hs.124049
- Hs.97530
- Hs.226955
- Clone.1339105
- Clone.1333841
- Clone.1334486
- Clone.682692
- Clone.704802
- Clone.2015

It is important to realized that some of the genes listed above will likely not be discriminatory because they express at a similar level across both subtypes. Therefore, as a second step to compiling a list of classifier genes we remove the candidate genes that occur in the highest supported patterns, but only if they are mapped to mapped to the same bin. The reason for doing this is that any gene that is supported by more tissues than occurs in either subtype is non-discriminatory. For example, if the expression of gene Y is mapped to the same bin across all the tissues, it would appear in every pattern. One way to identify such genes is to look at the highest supported patterns. In this case we combine all the genes that are supported by >24 tissues in $K_{max}$=10 and $K_{max}$=20 (see Table A.1 & Table A.2). Because these patterns are supported by more tissues than occur in either subgroup, these genes must be similarly expressed by both GCB and ABC cell lines. Ten such patterns are found in the results from $K_{max}$=10 and 7 patterns from the patterns discovered in $K_{max}$=20. When combined these genes are (* indicates gene appears in either of the above lists):

- JNK3
- PTK*
- MYBL1*
- RAD50*
- CD21*
- Hs.203866*
- Hs.75859*
- Hs.120716
- Hs.97530*

Notice that this list includes the 3 genes that appear as candidates for both subtypes (MYBL1, DCTD, Hs.75859). For deoxycytidylate deaminase the high support level of expression is “C” as it is for GCB, but the level is “D” (a higher amount of expression)

for ABC. For mybA the level of expression in the high support pattern is “A”, in the ABC pattern it is also ”A” but in the GCB pattern it is “B”. Hs.75859 is found to express at the same level in all three patterns. Therefore, the candidate genes for the ABC subtype are as listed above with the exception of myb A, and Hs.75859 and the candidate genes for the GCB subtype are as listed above with the exception of deoxycytidylate deaminase, and Hs.75859.

Throughout this section a number of tissues were found to cross-cluster with tissues from the other subtype. The worst of the cross-clusterings were {4, 24}, {21, 33}, and {21,41} because these patterns contained many co-expressing genes. However, the tissues 9, 13, 22, 24, 29, 31 and 38 were also found to cross-cluster frequently. Admittedly, these tissues cross-clustered in patterns with fewer genes, but did cluster with more tissues of the other subtype. When considering the reasons why these cross-clusterings occur, the first possibility to consider is that pattern discovery is simply not a good technique to use for this data. However, this seems unlikely because under certain ordering schemes pattern discovery does a very good job of identifying and clustering the subgroups in this data (see Table 4.26, lines $G_{min}$=8, $S_{min}$=10). Second, it can be seen that a number of the cross-clustering tissues identified throughout this section are outliers to the GCB and ACB subtypes (see Figure 4.4). This raises the possibility that the clustering method used by Alizadeh may have miss-clustered some of the tissues. For tissues 21, 22 and 24 this would likely be the case. However, this would not explain the cross-clustering of tissues 9, 29, 31, 33 and 38 because these tissues should represent the core of their respective subtypes. What this may indicate, however, is that there are further subtypes within these tissues that have not been identified. Indeed, since the original publication of this article by Alizadeh, it has come to light that there is a subtype of the ABC group called Type III [Rosenwald, ’02]. It could be that some of the tissues identified here as GCB might cluster better with a subgroup of ABC then they would with either ABC as a whole, or the GCB subgroup.

### 4.2.2 Pre-set value binning

Similarly to Section 4.1.4.4 this data was binned using a single set of values for every gene. The values used to delimit the bins were: -1.00, -0.25, +0.25, and +1.00. These values are used to form 5 bins in which to stratify the numeric data. The total execution time for pattern discovery on the Beowulf cluster was 42371 seconds (~12 hours). This section will examine the highest support patterns, the densest patterns, and the most discriminating patterns to estimate the minimum required values for support and length that good classifier patterns would need. As in the previous section, orderings of these patterns will be made and the ability of these patterns to classify subgroups of DLBCL will be analyzed. This will be followed by a discussion on any cross-classified tissues and a comparison made to the patterns found using Xmeans binning.

The highest support patterns have support values in the 17-28 range (see Table A.19). With only one exception (83% supported by GCB tissues) all the patterns were poor discriminators of DLBCL subtypes. The longest patterns contained from 172 to 256 genes (45.3% to 67.4% of the maximum possible) (see Appendix A, Table 20). With only 3 exceptions, all of these patterns were either 100% supported by ABC or GCB tissue types; with the largest proportion of patterns involving tissues 16-19. The three exceptions were supported by the following pairs of tissues: {1, 27}, {21, 33}, {11, 25}. When looking for patterns with perfect discrimination and ordered by support, it is found that only 3 of the top 20 are ABC type, and the rest are GCB (see Table A.21). All the patterns contain 1 to 3 genes and the highest support for both subtypes was 13. This means that for both ABC and GCB subtypes, approximately half of the samples could be classified by a single pattern containing 1 to 3 genes. This result would imply that good discrimination of the tissues should occur with as few as 1 to 3 genes and slightly higher values should be able to provide very good discrimination. Further, this implies that support values over 13 will likely not find patterns with good discrimination and definitely will not find patterns with perfect discrimination. However, support values below 13 should be able to consists entirely of one subtype or another. However, this was found to not be the case. Support values as low as 10 did not provide good discrimination, although $S_{min}$=5 did. So, in order to provide more data points for

changing $S_{min}$ at low values, $S_{min}$=7 was also used. In addition, a $G_{min}$ as high as 8 did not provide good discrimination either, so an ordering with a higher $G_{min}$ was also required. Therefore, the minimum genes and support values used for rankings were $G_{min}$=6, 8, and 12 and $S_{min}$=5, 7 and10. More complete results for pattern discovery with data from $G_{min}$=8, $G_{min}$=12, $S_{min}$=5, and $S_{min}$=7 is available in Table A.22 through Table A.27, respectively.

Unlike in the Xmeans binned data, the minimum gene criteria here included a fairly equal number of patterns that clustered both ABC and GCB subtypes (see Table A.22 & Table A.23). In Table 4.28 the number of patterns that successfully classify either subtype are listed. As can be seen, these patterns do a very poor job of discriminating one class from another. This is due to the cross clustering of 5 tissues {24, 25, 33, 38 and 39} in predominately GCB supported patterns and 5 tissues {2, 11, 18, 19 and 22} in predominately ABC supported patterns. The two most common genes in these cross-clusters were RAD50 and TTG2.

**Table 4.28. The number of patterns that successfully classify, at the given percentage, either the ABC or GCB subtype for the listed minimum gene requirement**

| Criteria | 80% | 90% | 100% |
|---|---|---|---|
| $G_{min}$=6 | 2 | 1 | 1 |
| $G_{min}$=8 | 6 | 2 | 2 |
| $G_{min}$=12 | 6 | 4 | 4 |

Unlike with the minimum gene requirement, the minimum support requirement did a much better job of selecting good patterns. This can be seen in Table 4.29 where, for GCB patterns the $S_{min}$=5 ordering produces 17/20 patterns with perfect discrimination (see Table A.24), compared to the $G_{min}$=12 ordering which only produces 4/20 patterns with perfect discrimination. However, the top 20 patterns for each $S_{min}$ were dominated by GCB tissues so a second ordering which only included patterns that were at least 50% ABC tissues was also constructed (see Table A.25 & Table A.27). The results of these orderings can be seen in Table 4.29. Although the top 20 patterns with $S_{min}$=5 did a good

job of predicting the GCB subtype, they required between 28 and 32 genes to do so. At $S_{min}$=7 only 11 to 13 genes were required to make the classifications, but there is a deterioration in discrimination (10 vs. 17 perfect discriminators). Unfortunately, as can be seen in Table 4.29, these patterns do a very poor job of classifying the ABC subtype. Even the poor results at $S_{min}$=5 for the ABC subgroup required 28 genes to accomplish.

**Table 4.29. The number of patterns that successfully classify, at the given percentage, the ABC and GCB subtypes for the listed minimum support requirement.**

| **Criteria** | At least 50% ABC | | | At least 50% GCB | | |
|---|---|---|---|---|---|---|
| | **80%** | **90%** | **100%** | **80%** | **90%** | **100%** |
| $S_{min}$=5 | 5 | 3 | 3 | 19 | 17 | 17 |
| $S_{min}$=7 | 0 | 0 | 0 | 13 | 10 | 10 |
| $S_{min}$=10 | 0 | 0 | 0 | 2 | 1 | 0 |

The results from this section stand in stark contrast to the previous section where the ABC class dominated the top 20 patterns. Here, GCB patterns seem to be the densest and most discriminating. Generally speaking, however, Xmeans binning out preformed pre-set value in a number of ways. For instance the Xmeans binning scheme finds a higher support pattern with perfect discrimination than does the pre-set value technique (18 vs. 13 tissues), and although the pre-set value technique did find patterns with more genes in them (67% vs. 41%), this did not translate into more patterns with good discriminating power. For example, the Xmeans binned patterns can produce much better discrimination using 4 genes than can the pre-set value patterns using 8. Further, Xmeans binning produces more patterns that are perfectly, or nea*r*-perfectly composed of just one subtype of another at any particular support level.

Composing a succinct list of candidate classifier genes from the data is this section would be difficult because very few of the orderings produced good classifying patterns and the good orderings which did produced good classifying patterns contain many genes. Because no sorting criteria did an especially good job on finding classifier patterns for the ABC group, no candidate genes will be proposed for identifying the ABC subtype.

Instead of using the top 20 patterns from $S_{min}$=7 (which is used as a compromise between accuracy of classification and the number of genes to achieve this classification) to compose a list of candidate genes for the GCB subtype, only the top 5 patterns will be used (see Table A.26). Doing this yields the following candidate genes for identifying the GCB subtype of DLBCL (* indicates gene also occurs in Xmeans binned list):

- KIAA1037
- PTP4A (Protein tyrosine phosphatase type IVA)
- ALU subfamily SB
- CD10*
- KIAA0805
- KCNA3 (Potassium voltage gated channel shaker related subfamily member 3)
- RAD50
- WASPIP* (WIP HS PRPL2 WASP interacting protein)
- TTG2
- SRPK2
- IL10RB (Cytokine receptor family II member 4)
- CD44
- CSNK1G2* (casein kinase 1)
- BCL2*
- PTP1B
- ALU subfamily SQ
- Hs.59368
- Hs.192047
- Hs.123294
- Hs.97275
- Hs.190288*
- Hs.120716*
- Hs.224323
- Hs.88102
- Hs.192738
- Clone.825199
- Clone.814651
- Clone.2017
- Clone.713158
- Clone.1371532
- Clone.1351325
- Clone.1288046

Normally these candidate genes would be pruned of any genes that occurred in a pattern that definitely had enough support to bridge both subclasses of DLBCL. However, in this data set only one pattern has enough support to do this. As an alternative to that approach, the top 10 patterns with the highest support will be used to form a list of cross-

clustering genes (see Table A.19). Doing so yields this set of genes (those marked with a * occur in the above list):

- JMJD1B
- SLCO3A1
- IL2RB
- BCL2*
- Hs.203866
- Hs.190288*
- Hs.4766
- Hs.87589
- Hs.123318

Subtracting genes in the second list from those in the first list yields the following candidate genes:

- KIAA1037
- PTP4A (Protein tyrosine phosphatase type IVA)
- ALU subfamily SB
- CD10*
- KIAA0805
- KCNA3 (Potassium voltage gated channel shaker related subfamily member 3)
- RAD50
- WASPIP* (WIP HS PRPL2 WASP interacting protein)
- TTG2
- SRPK2
- IL10RB (Cytokine receptor family II member 4)
- CD44
- CSNK1G2* (casein kinase 1)
- PTP1B
- ALU subfamily SQ
- Hs.59368
- Hs.192047
- Hs.123294
- Hs.97275
- Hs.120716*
- Hs.224323
- Hs.88102
- Hs.192738
- Clone.825199
- Clone.814651
- Clone.2017
- Clone.713158
- Clone.1371532
- Clone.1351325
- Clone.1288046

Of the 32 genes determined to support the GCB subtype, 6 are shared with the list from Section 4.2.1 and two of these (BCL2 and Hs.190288) were pruned from both result sets. Interestingly JMJD1B is considered non-discriminatory in this data set but was a candidate classifier gene in the previous section.

### 4.2.3 Genes@Work Approximation

Work on the Genes@Work approximation immediately presented two major difficulties. First, to approximate even small delta values in Genes@Work, many characters are needed. This leaves very few characters to use as break-characters. Second, because of the need to find similar patterns (see Section 2.5.2), pattern discovery takes a very long time and cannot finish in less than 24 hours on the Beowulf cluster with $K$=2. With higher values of support it is possible to get pattern discovery to complete, but the results were poor. Many of the patterns were not similar to those found by Genes@Work. The latter's patterns being superior to the approximation; generally having more genes for any particular level of support and finding a pattern with higher support than any pattern found by the approximation. Although it would be likely possible to improve the results of the approximation, time restrictions limited further development of it.

### 4.2.4 Classification with Genes@Work

Unlike the pattern discovery and "blind" clustering methods used above, Genes@Work takes a supervised clustering approach to the classification problem. That is, Genes@Work requires each array in the data set to be labeled with its class. It then uses one class (the training data) to help stratify the data in the other class (test data). Genes@Work then uses pattern discovery to find similar patterns in the test data. Because of the way it stratifies the data Genes@Work can estimate the probability of the patterns it has found of being discovered by chance. It uses this scoring system to evaluate the quality of a pattern.

Genes@Work's approach to microarray analysis impacts the evaluation of pattern quality in three ways. First, because Genes@Work only finds patterns in a single class of microarrays, all its patterns perfectly segregate the two classes. This prevents evaluating the quality of the patterns based on how well they discriminate between ABC and GCB tissue types. However, it is still possible to compare the quality of Genes@Work's results to those of Teiresias, using the other two binning methods, in terms of the length and support of the discovered patterns. Also, it is possible to compose a list of candidate genes from Genes@Work's patterns and compare this gene list to the genes obtained by the Xmeans and pre-set value techniques. Second, because Genes@Work removes correlated patterns, not every pattern discovered will be listed in the final results. It is believed that the most important patterns will still remain in the result set. However, to offset the fact that some patterns have been removed, only the top 10 patterns for each ordering of support, length and probability will be considered. Lastly, the effects of using Genes@Work's scoring system are investigated.

Genes@Work was executed twice on this data set. The first time using the GCB tissues as the test data, and the second time using them as the training data.

Discovering patterns in the GCB data took 117 seconds. When these results are ordered by support (see Table 4.30), a single pattern of support 21, captures every GCB tissue sample except 21, 22, and 23. No other pattern in the top 20 patterns listed by support supports any of these 3 tissues. The pattern uses the following 3 genes to cluster the tissues: ALU subfamily SB, Hs.224323, and Hs.136345.

**Table 4.30. Top 10 patterns discovered by Genes@Work when ordered by support**

| Rank | Probability ($log_{10}$) | Supporting Tissues | Genes |
|---|---|---|---|
| 1 | -87 | 21 | Alu subfamily SB, Hs.224323, Hs.136345, clone.825199 |
| 2 | -102 | 20 | Hs.120716, ALU subfamily SB, Hs.224323, Hs.136345, clone.825199 |
| 3 | -102 | 20 | Hs.136345, Hs.224323, ALU subfamily SB, clone.825199, ALU subfamily SB |
| 4 | -95 | 19 | Hs.136345, clone.825199, ALU subfamily SB, Hs.224323, Hs.105261 |
| 5 | -116 | 19 | Hs.120716, Hs.136345, Hs.224323, ALU subfamily SB, clone.825199, ALU subfamily SB |
| 6 | -108 | 18 | Hs.136345, clone.825199, Hs.224323, ALU subfamily SB, Hs.105261, ALU subfamily SB |
| 7 | -108 | 18 | Hs.120716, Hs.136345, clone.825199, ALU subfamily SB, Hs.224323, Hs.105261, |
| 8 | -108 | 18 | Hs.136345, clone.825199, Hs.224323, ALU subfamily SB, mybA, ALU subfamily SB |
| 9 | -82 | 17 | Hs.136345, clone.825199, Hs.224323, ALU subfamily SB, Hs.222808 |
| 10 | -117 | 17 | Hs.120716, HJs.136345, clone.825199, ALU subfamily SB, Hs.224323, mybA, ALU subfamily SB |

Ordering the patterns by their length (see Table 4.31) shows that two tissues, 16 and 17, are very strongly related with 254 genes (which closely matches the 256-gene longest pattern found in Section 4.2.2). The next 9 patterns all have support 3 or 4 and include tissues 16 and 17 in their support list. The other supported tissues are: 5, 9, 10, 12, 13, and 14. Each pattern is composed of 102 to 156 genes.

**Table 4.31. Top 10 patterns discovered by Genes@Work when ordered by length**

| Rank | Probability ($log_{10}$) | # of genes | Supporting Tissues |
|---|---|---|---|
| 1 | -156 | 254 | 16, 17 |
| 2 | -221 | 156 | 12, 16, 17 |
| 3 | -184 | 137 | 15, 16, 17 |
| 4 | -180 | 135 | 13, 16, 17 |
| 5 | -147 | 117 | 10, 16, 17 |
| 6 | -143 | 115 | 9, 16, 17 |
| 7 | -140 | 113 | 14, 16, 17 |
| 8 | -131 | 108 | 5, 16, 17 |
| 9 | -243 | 103 | 12, 15, 16, 17 |
| 10 | -240 | 102 | 12, 13, 16, 17 |

When using Gene@Work's own ordering system we are presented with the following top 10 patterns listed in Table 4.32.

**Table 4.32. The top 10 patterns discovered by Genes@Work when ordered by probability**

| Rank | Probability ($log_{10}$) | # of genes | Supporting Tissues |
|---|---|---|---|
| 1 | -249 | 74 | 12, 13, 15, 16, 17 |
| 2 | -242 | 103 | 12, 15, 16, 17 |
| 3 | -239 | 102 | 12, 13, 16, 17 |
| 4 | -238 | 54 | 12, 13, 14, 15, 16, 17 |
| 5 | -230 | 69 | 9, 12, 15, 16, 17 |
| 6 | -230 | 69 | 12, 14, 15, 16,17 |
| 7 | -230 | 69 | 10, 12, 15,16, 17 |
| 8 | -229 | 52 | 10, 12, 13, 15, 16, 17 |
| 9 | -225 | 41 | 10 12, 13, 14, 15 , 16, 17 |
| 10 | -224 | 51 | 10, 12, 14, 16, 17 |

There are two interesting things about these results. First, all the patterns are built around the tissues {16, 17}. These agrees with the results from Cluster (see Figure 4.4), which

tightly clusters these two tissues before than others in the GCB subtype. Second, the smallest number of genes in any of the above patterns is 41. Although these patterns may be the least probable to have occurred by chance, from a clinical stand point, 41 is far too many genes to construct a classifier upon.

Because of the high number of genes in Genes@Work listing, the only effective way to construct a list of candidate classifier genes is from the highest support list. Because a single pattern captures nearly all the tissues, this means using only one pattern to compose the list of candidate genes. This one pattern is also significant because it occurs with the highest support of any GCB pattern discovered by any technique. Although three tissues {21-23} are not in this pattern's support list, these tissues are also least correlated tissues (as determined by Cluster) to the GCB group and therefore it seems reasonable that they might not be a part of the GCB group. Of the three genes it identifies as classifiers {ALU subfamily SB, Hs.224323, and Hs.136345}, only ALU subfamily SB was also determined by Xmeans to be a classifier gene. However, the pre-set value technique agrees with two of the genes (ALU subfamily SB and Hs.224323). Interestingly, Hs.224323 is a candidate classifier gene for the ABC subtype in Section 4.2.1. The gene appears in both lists because in the ABC subtype the gene is moderately down-expressed and in the GCB subtype it is strongly up-expressed.

Having examined the GCB patterns and composed a candidate list of genes the focus now changes to the ABC subclass of DLBCL. Using Genes@Work to discover patterns in the ABC cell lines takes 74 seconds. The two highest support patterns both have a support of 17 (which compares well to the highest support pattern found with the Xmeans method which has 18 supporting tissues). Between the two patterns they support every ABC tissue and therefore are the only patterns required to fully define the ABC subtype. Because of this the genes in both patterns forms the candidate gene list. The two patterns are shown in Table 4.33.

**Table 4.33. Candidate classifier genes for ABC subtype discovered by Genes@Work**

| Rank | Supporting Tissues | Genes |
|---|---|---|
| 1 | 24, 25, 27-42, 43, 45 | Clone 1353015, JAW1 |
| 2 | 24, 26, 29, 31, 32, 35-47 | Deoxycytidyle deaminase, ALU subfamily SB |

Neither gene in the first pattern was nominated as a classifier gene in either of the other two techniques. However, both genes in the second pattern were. ALU subfamily SB appeared as a classifier by the Xmeans method. As mentioned above, it also appears as a classifier for the GCB subtype. Deoxycytidyle deaminase was nominated by the Xmeans method as a candidate gene as well.

When ordered by length we see that the longest pattern contains 104 genes. Interestingly, all the uncorrelated patterns have at least 3 supporting tissues which may account for the smaller number of genes in the longest pattern, especially when compared to the longest pattern discovered in the GCB subtype. The top ten patterns by length are listed in Table 4.34.

**Table 4.34. The top 10 patterns sorted by length, discovered by Genes@Work for the ABC DLBCL subtype.**

| Rank | Probability ($log_{10}$) | # of genes | Supporting Tissues |
|---|---|---|---|
| 1 | -124 | 104 | 35, 42, 43 |
| 2 | -101 | 90 | 36, 42, 43 |
| 3 | -94 | 86 | 35, 36, 43 |
| 4 | -88 | 82 | 35, 36, 42 |
| 5 | -86 | 81 | 36, 39, 43 |
| 6 | -85 | 80 | 35, 39, 43 |
| 7 | -82 | 78 | 36, 39, 32 |
| 8 | -80 | 77 | 35, 36, 39 |
| 9 | -76 | 74 | 39, 32, 32 |
| 10 | -73 | 72 | 40, 42, 43 |

In the above table it can be seen that a few tissues provide the core of the ABC subtype. Although not as tightly knit at the core of the GCB group, tissues 39, 42 and 43 correlate highly to other tissues in the group.

The patterns, sorted by probability are presented in Table 4.35. Again, we see that patterns with low support and a high number of genes are favored.

**Table 4.35. The top 10 patterns sorted by probability, discovered by Genes@Work for the ABC DLBCL subtype.**

| Rank | Probability ($log_{10}$) | # of genes | Supporting Tissues |
|---|---|---|---|
| 1 | -132 | 63 | 35, 36, 42, 43 |
| 2 | -127 | 31 | 35, 36, 40, 42, 43, |
| 3 | -124 | 104 | 35, 42, 43 |
| 4 | -122 | 40 | 35, 36, 39, 42, 43 |
| 5 | -118 | 39 | 35, 36, 39, 40, 43 |
| 6 | -111 | 55 | 35, 40, 42, 43 |
| 7 | -111 | 37 | 35, 39, 40, 42, 43 |
| 8 | -109 | 54 | 35, 36, 39, 43 |
| 9 | -108 | 36 | 35, 36, 39, 40, 42 |
| 10 | -107 | 53 | 35, 39, 42, 43 |

Generally, Genes@Work does an excellent job classifying this data. This is shown by three things. First, it finds patterns with higher, or nearly as high support as the other two methods tested. Second, it finds patterns that are as long or longer than previous discovered, for both GCB and ABC subtypes. Third, the genes in the highest support patterns (the candidate genes) agree reasonably well with the candidate genes from the other two methods. Indeed, the patterns discovered by Genes@Work require fewer genes to correctly classify more tissue samples than any patterns discovered by Teiresias. For this reason Genes@Work provides better candidate genes than either of the other two binning techniques.

It should not be surprising, however, that Genes@Work discovers more discriminating patterns than Teiresias. Genes@Work is a supervised pattern discovery application, or rather, its binning technique takes advantage of the class labels provided with the data. Teiresias, on the other hand, was used on data that was binned without knowledge of which array belonged to which class. This is likely why, although the other two techniques provided a similar list of candidate classifier genes, Genes@Work seemed to

have a much smaller, concise list, whereas Xmeans clustering and pre-set value had much larger lists where individual patterns had less support. This does not necessarily mean that the genes listed in the other two techniques are wrong, or cannot differentiate a subgroup of DLBCL, but they do not separate the ABC and GCB subtypes as efficiently as the genes found by Genes@Work under its own binning scheme.

Genes@Work also seems to confirm that some tissues have been incorrectly classified by cluster. This is evidenced by the pattern used to classify the GCB subtype. That pattern was supported by every tissue sample except for three. Those three tissues (21-23) have already been shown, by patterns discovered by Teiresias, to cross-cluster with the ABC subtype. Their failure to form a strong pattern with the rest of the GCB tissues helps support the hypothesis that they have been incorrectly classified.

Despite its successes there are three areas where Genes@Work does not perform very well as a classifier of microarray data. First, Genes@Work assumes that the class labels are correct. In other fields where classification is used, this may be a very reasonable assumption. However, with DLBCL, class labels may be incorrect. For instance, in every technique above a number of tissues either cross-classified with the other subtype, or in the case of Genes@Work, failed to classify well with its own class. These tissues are almost certainly mis-classified. Mislabeled tissues highlight an advantage of taking a clustering, as opposed to classification approach to deciding which tissues belong together. When clustering, it is possible to find data that has been mis-classified; when using classification approaches mis-classified data may go undetected. Second, Genes@Work makes the "two class assumption": that in any data set there are only two classes. Admittedly this is a failing of almost all classification techniques. However, unlike many other classification techniques, Genes@Work effectively forms a clustering of each class by discovering patterns in it; each pattern can be seen as its own cluster. Thus, Genes@Work should be able to determine subclasses that exist within the two main classes. Third, Genes@Work's probability calculation favors many genes with low support, over less genes with higher support. Although this may be the correct approach mathematically, it fails from a clinical perspective where there is a cost associated with

using more genes in the classifier. Admittedly, a user can set a minimum support level for any pattern (which usually reduces the number of genes in the pattern), however it would be ideal if Genes@Work could come up with a better listing of patterns using even low values for minimum support.

# 5 Discussion and Future work

The major goals of this project were three-fold. The first goal was to evaluate three different pattern discovery methods – Teiresias, Splash and Genes@Work – in terms of memory and execution time efficiency. The second goal was to evaluate, based on their ability to form biologically meaningful patterns, three binning techniques – the sign-of-the-derivative, pre-set value and Xmeans – on yeast cell cycle data . Finally, this work explored the ability of three binning techniques – Xmeans, pre-set value and the Genes@Work method – to find distinct subgroups of disease in diffuse large B-cell lymphoma (DLBCL) patient tissue samples. These results were compared on the basis of the length (number of genes) and support of the patterns, and the pattern's ability to distinguish the two subclasses (ABC and GCB) of DLBCL. To complete the above three goals, two secondary objectives were also achieved in this work. The first, was to formulate an ordering scheme, based on pattern length and support, that would rank biologically significant patterns highly and the second was to compare the results of pattern discovery to the results from Cluster.

The timing and space results for Teiresias, Splash and Genes@Work were very clear cut. The execution time for all three algorithms was linear with respect to the number of patterns discovered. However, the slope of the linear relationship was very different for each of the three programs. Except for very small result sets, where Teiresias and Splash had near identical run times, Teiresias was two to five times faster than Splash, and Splash was ten to 100 times faster than Genes@Work. Further, as more patterns were discovered the time difference between applications becomes more pronounced.

The memory requirements for each application varied. Teiresias' memory usage (for any one data set) increased exponentially with decreasing $K$, even if only a few more patterns

were discovered. For equal values of $K$ across different sized data sets, the memory requirement increased linearly with the number of patterns discovered. Unlike Teiresias, Splash's memory requirements increased linearly with respect to the number of patterns discovered regardless of the value of $K$. Genes@Work's memory usage was generally linear in the number of patterns discovered with some variability in memory consumption which was most likely due to the Java virtual machine memory allocation. For the data sets explored here, Splash uses 2 to 3 times less memory than Teiresias, and Teiresias uses 2 to 3 times less memory than Genes@Work. Further, as the number of patterns increases, these differences become more pronounced.

From these results it can be seen that Teiresias is significantly faster, but less memory efficient than Splash, Splash is slower but more memory efficient than Teiresias, and that Genes@Work is slower and less memory efficient than either of the other two programs. Deciding which of the latter two application is the best is not a simple task. Memory efficiency is not an insignificant matter in pattern discovery. In almost all cases, when pattern discovered failed with Teiresias it was because Teiresias ran out of available memory space. However, the speed difference between Teiresias and Splash out-weighs the difference in memory efficiency; many executions on 612 genes of the yeast cell cycle simply would not finish with Splash under 72 hours, but would finish with Teiresias in 8 hours, even though the execution may use all available memory. Generally, pattern discovery on a large data set simply would not finish with Splash in a reasonable amount of time. Provided with enough memory, Teiresias can finish pattern discovery on very large data sets in a small amount of time. Further, providing Teiresias with more memory has become easier and cheaper in recent years with the advent of Beowulf clusters and cheap 64-bit "desktop" computers.

Despite its slowness and memory inefficiency, Genes@Work has many features that might make it good for pattern discovery. First, Genes@Work only discovers aligned patterns. Aligned patterns account for only 25% of the total patterns discovered by Teiresias and Splash in this work. This means Splash and Teiresias do four times more work than they need to in the context of this thesis. Further, using break characters to

help remove unaligned patterns wastes characters that could be used for data bins. Second, Genes@Work uses a global density requirement whereas Splash and Teiresias use local ones. In any application where the elements of a stream have no order, a global density requirement makes more sense than a local one because there is no reason to believe patterns will consists of elements in close proximity. For example, in Section 4.2.4, 2 genes (ALU subfamily SB and Deoxycytidylate deaminase) were found to be significant; these are separated by over 200 genes in the original data set (not shown), but they equally could have been further apart or closer together; their position in the data is arbitrary and a global density requirement better captures this fact than a local one. The final beneficial feature of Genes@Work is that it allows the researcher to visualize the data through a GUI and to use other clustering methods to analyze input data. Generally, when trying to cluster the microarrays (data columns) together, Genes@Work is a very good choice, but improvements could be made by increasing efficiency (time and memory), allowing the option of command-line execution (for automation), and decoupling the data-binning from the pattern finding.

It should be noted that many of Genes@Work's features could be added to Teiresias (or Splash). Although implementing a global density requirement in Teiresias would be difficult, implementing a system by which only aligned patterns are discovered would be relatively easy. It would also be possible to make a fully-featured GUI for Teiresias without having to change the core algorithm in any way.

Three methods of data stratification were attempted with the yeast cell-cycle data: the sign-of-the-derivative, the pre-set value, and the Xmeans technique. The quality of these techniques is based primarily on their capacity to capture biologically meaningful relationships between the data and to a lesser extent, their ability to not produce a relatively large number of patterns and thus overly increase execution time. Certainly, if two techniques find the same, or similar biologically significant patterns, then the technique in which pattern discovery was fastest is better. With regards to the second criterion, the sign-of-the-derivative technique was considerably faster than either of the other two techniques, with $\delta = 0.2$. The pre-set value data eventually required the use of

the Beowulf cluster to complete execution, and the Xmeans technique not only required the Beowulf cluster, but also an increase in the density requirement of the patterns for pattern discovery to complete. For many of the patterns, the biological significance of the clusters was reasonably easy to deduce. The most prominent of all the patterns was the histone pattern which contained 8 histone genes, and occurred in some form across multiple parameter settings. However, genes involved in rRNA-processing and protein replication were also discovered. One of the most interesting patterns was the bud-neck group, which involve a set of genes that are not functionally related, but rather co-localized to the same place in the cell.

Two of the discovered patterns are of particular interest because they would imply a biological function for genes with no functional annotation. The first pattern (see Table 4.6) consists of the following genes: NOP7, HCA4, MRT4, EBP2, MRD1, DIM1, NOP6, YMR269W and YPL146C. All the genes except the last two are rRNA processing genes. This pattern would suggest that YMR269W and YPL146C are involved in the rRNA processing and rRNA. Further, it is suspected [SGD] that YMR269W is involved in protein synthesis. This reinforces the latter conclusion because rRNA is required to form ribosomes, which are used to produce proteins. The second important pattern (see Table 4.15) is supported by the following genes: HHT1, HTB1, HTA1, HHF2, HHT2, HHO1, HTA2 and SVS1. The first 7 genes are all histone genes. The last gene is involved in vanadate resistance. This suggests a possible SVS1-histone interaction and a role for histones in vanadate toxicity. Despite the strong evidence for roles for these unknown genes, it is important to view this data as generating a hypothesis rather than confirming a hypothesis. The only way to confirm the roles of SVS1, YMR269W and YPL146C would be to perform ("wet-lab") biological experiments.

Many of the same biologically significant patterns occurred in both the pre-set value binned data and the sign-of-the-derivative data. Xmeans binned data found very few patterns of biological significance with the exception of two galactose processing genes that were clustered together with a long pattern. Although all techniques showed some degree of success, of the three, the sign-of-the-derivative technique was the best. It

discovered six of the ten most biologically significant patterns, where as pre-set value discovered only three and Kmeans resulted in just one significant pattern. Further, many of the patterns discovered by the pre-set value and Kmeans techniques were also discovered using sign-of-the-derivative data.

In an attempt to select the biologically significant patterns from the millions discovered, the patterns were ordered by length. Ordering the patterns by length, generally, seemed to be a good way of ranking them. The majority of the top-ranked patterns had biological significance. Genes@Work was also used to sort its own patterns by probability. With one exception (an rRNA-processing group) Genes@Work did a poor job of ordering the patterns. Few of the top ranked patterns held any consistent biological function across all the genes in the pattern. Instead, only subsets of the genes seemed to have biological meaning. It might be the case that if Genes@Work were to favor longer, lower support patterns, it would be more likely that these patterns would have biological meaning.

After the biological significance of the discovered patterns was investigated, the gene groups that support each of the most biologically significant patterns were compared to the gene clusters formed by Cluster. In almost all cases, Cluster found similar (although not exactly the same) groups to those formed by pattern discovery. As an example, pattern discovery grouped together SVS1 with 8 histone genes but in Cluster, SVS1 was not found with the histone genes. Instead, Cluster placed the histone genes in a better (the biological role was more clear) group with two more histone genes. A similar result occurred a number of times; for a group of genes with some biological role, Cluster would find a larger group that shared the same role. In this aspect, Cluster's results were better than those of pattern discovery.

However, Cluster is inferior to pattern discovery in a number of ways as well. First, pattern discovery would often group genes that had some specific biological role. However, when the group was investigated in Cluster's results, there would often be extra genes within the cluster that would "dilute" the specific biological role of the genes to a more general one. Thus a group of genes determined to be regulated by SBF by

pattern discovery falls into the more general role of cell-proliferation in Cluster. It is felt that Cluster would often lose the finer relationships between genes that can be found by pattern discovery. This no doubt stems from hierarchical clustering's inability to allow genes to be part of more than one cluster. As is often the case, a single gene may be involved in more than one biological role. Such a gene can be a part of many relationships in pattern discovery, but can only appear in a single relationship in Cluster. Generally speaking, Cluster seemed to out-perform pattern discovery on this data. Although its clusters are not superior to pattern discovery in every way, Cluster generally does a good job of finding the same relationships as found by pattern discovery and occasionally manages to refine them. Cluster's superiority is compounded by its execution times. Cluster requires approximately 60 seconds to finish clustering the yeast cell cycle data set whereas pattern discovery takes hours. Certainly, for quick, generally good results, Cluster is a good choice for this data set.

Considering the results of the comparison with Cluster, there are a number of ways in which pattern discovery for this kind of data may be improved. The success of Cluster seemed to be based on genes generally expressing at the same values across many time points. Thus a technique that stratifies the data to fewer bins, but requires the patterns to be more dense, may find better results. In addition to using fewer bins, similar patterns could also be searched for. In such an approach, bins that represent adjacent ranges of values would be allowed to form a pattern. This would mean 'A' and 'B', 'B' and 'C', 'C and 'D' and so on, could form patterns. This has a similar effect to using fewer bins. Both of these approaches would increase execution time for pattern discovery. However, by requiring patterns to be denser and to have higher support, these time increases could be allayed.

Compared to the other binning techniques the results from Xmeans were disappointing. One possibility for improving the results with Xmeans would be to use it to stratify the data array by array. That is, instead of using Xmeans on a gene-by-gene basis, it could instead be used on an array-by-array basis. Then pattern discovery could take place normally. By using Xmeans to bin "in the opposite direction" of pattern discovery we

alleviate the need to compress the bins for the same reason that it did not have to be done with the lymphoma data (see Section 3.2.6). This approach has the added advantage of being an uncommon way to analyze this type of data and thus might be able to show new types of patterns.

In addition to the yeast cell cycle, pattern discovery was also used on the classification/clustering problem of DLBCL. Teiresias was used to discover patterns in two different stratifications of this data stratifications using Xmeans and pre-set value. Genes@Work, which uses its own method of data stratification, was also used on this data. When considering the quality of these techniques, both pre-set value and Genes@Work found longer, higher support patterns in the GCB subtype of DLBCL than in the ABC subtype. Xmeans, however, found better patterns in the ABC subtype. Compared to the pre-set value technique, Xmeans was able to find many more patterns that could perfectly or near perfectly discern between the two subtypes using the same number of genes (or less), or using patterns with higher minimum support. Because of this, Xmeans is deemed the better stratification technique of the two. However, Genes@Work found patterns that were better still. For both GCB and ABC, one or two patterns, with 2 or 3 genes each, could capture all or nearly all the tissues in a subtype.

For the GCB subtype, 21 of 24 tissues formed a pattern composed of 3 genes. Of the three tissues, two were found by the other techniques to cross cluster with ABC tissues. Further, the three tissues are also the least correlated with the rest of the GCB subtype. Thus it seems very likely that they do not belong with the GCB subtype.

For the ABC subtype, Genes@Work discovered 4 genes in two patterns that seem to successfully separate the two subtypes: Clone 1353015, JAW1 and Deoxycytidyle deaminase, ALU subfamily SB. These may represent two distinct subtypes in the ABC class. Both pre-set value and Xmeans identify both Deoxycytidyle deaminase, ALU subfamily SB as candidate classifiers of ABC but neither pick out the other genes as being good classifiers. For the GCB subtype, Genes@Work discovers 3 genes that seem to define the cluster: ALU subfamily SB, Hs.224323, and Hs.136345. Again, both

Xmeans and pre-set value techniques agree with 2 of the three genes (ALU subfamily SB, and Hs.224323). Interestingly ALU subfamily SB appears in both lists. It may be concluded then, that this gene provides the best discretion of all the genes for determining subtypes in DLBCL.

Cross-clustering was a common occurrence for both Xmeans and pre-set value binned data. The most frequent cross-clustering tissue samples were 1, 2, 21,24 and 25. In all cases, these samples represent outliers to the central core of their respective subclasses. From this it may be concluded that the samples have likely been mis-classified. This conclusion would not have been possible to reach using the results from Cluster. Because of this any future work in clustering/classification should not use Cluster as the definitive clustering technique and should instead use other methods to determine the correct classification of the samples. On the other hand, pattern discovery can discover mis-classifications because pattern discovery does not force sample to be in a single group, and it allows samples to belong to multiple groups. Because of this, pattern discovery can leave certain samples ungrouped and allow certain other samples to be part of both subgroups. This allows the discovery of out-liers and "troublesome" samples.

In addition, pattern discovery has other advantages over Cluster. First, it allows for the possibility of discovering new subtypes of the disease and, two, when a new subtype is recognized, it can instantly identify which genes contribute to defining the cluster. Although it would be possible to achieve some of this with hierarchical clustering, none of the required analysis tools have been built into Cluster.

Pattern discovery is an excellent tool for classification/clustering. Even with a very simple approach to the problem pattern discovery managed to identify probable errors in the classification/clustering made by Cluster. The stratification technique of Genes@Work illustrates the advantage of using a machine-learning approach (training data) over discretization methods which do not use training data (Kmeans, pre-set value). Further, Genes@Work's approach to classification/clustering could allow for the detection of mis-classified samples, a problem in many biological fields. Allowing

Genes@Work to discover mis-classified elements would be a marked improvement over many of the classification techniques commonly used in microarray analysis. Although not as good as Genes@Work's technique, Xmeans stratification showed marked improvement over naïve pre-set value binning on DLBCL data. Further, the Xmeans binning technique might be improved by allowing the discovery of similar patterns. Although this would increase the execution time of Teiresias, more conservative parameters could also be used reduce execution time. For instance the results from this section indicate that the minimum support $K$ could be set to 1/6 or 1/10 of the number of streams and still yield good results. Further, a global density requirement (currently only available in Genes@Work) could be set between 2 and 8 and provide good results as well.

Another way of improving the results of pattern discovery would involve devising a better method of picking patterns to discover subtypes. In this work, the identification of the DLBCL subtypes relied, simplistically, on using a single pattern (or a collection of patterns each considered individually) to separate the two subclasses. Another approach that could be explored would be to cluster patterns together based on their correlation to one another. This seems like a good approach because the majority of the top 20 patterns in any ordering were highly correlated to one another. For instance, the top 5 patterns in $K_{max}$=10, $S_{min}$=5 have the following tissues (all of which discriminate perfectly) in their support list: {34, 37, 39, 40, 41}, {34, 36, 39, 42, 43}, {32, 34, 36, 39, 43}, {34, 35, 39, 42, 43}, {32, 34, 39, 40, 41}. This high-correlation situation is quite similar to that encountered with the yeast cell-cycle patterns. One approach to clustering this data might be to use a number of low support, highly discriminatory patterns and merge them into a single cluster, based on how the patterns are correlated. For example, merging the 5 patterns listed above would yield a single cluster containing tissues {32, 34, 35, 36, 37, 39, 40, 41, 42, 43}. Note that this cluster has double the support of any of the patterns that it is composed from, and yet still discriminates perfectly.

Although pattern discovery shows great promise in the field of classification and clustering, in order to improve on its success four changes in the way pattern discovery

algorithm authors approach patterns must be made. First, it should be recognized that pattern discovery is a form of clustering. Second, microarray analysis, as it stands now, almost exclusively involves aligned patterns only and pattern discovery algorithms should reflect that. Third, aligned patterns really cluster both genes and arrays. Lastly, microarray analysis packages that use pattern discovery should take advantage of more of the information contained in the resulting patterns.

Although not often referred to as such, pattern discovery is a clustering technique. It brings together different streams based on relationship contained in the template of the pattern. Specifically, Genes@Work should not be thought of as a classification technique but as a clustering technique with hybrid-classification stratification aspects. Further research with Teiresias in microarray analysis should draw upon knowledge in the clustering field to help form meaningful patterns.

Admittedly, time-series data could contain unaligned patterns, but the majority of microarray clustering (and all microarray classification) studies involve aligned patterns only. Genes@Work only searches for aligned patterns yet Teiresias does not have such an option. If Teiresias should be used for microarray analysis it should have the ability to only search for aligned patterns only. Further, when only aligned patterns are considered, patterns gain a three important new properties. Firstly, all data points in a stream are independent of all others. This has a practical use when running pattern discovery across multiple computers; the input data is now divisible across streams. Secondly, all support lists need only contain the streams that contain the pattern, because aligned patterns can be thought of always starting at position 0 in a stream. Thirdly, a pattern can be uniquely identified by its support list. This is because only one pattern can exist between any unique set of streams. This could allow for a very efficient hashing scheme (using the combined stream identifiers to uniquely identify the pattern) , which could be used to run pattern discovery in parallel.

It should be recognized that an aligned pattern is not just a clustering of streams (arrays) but also of literals (genes). The only difference between the two forms of pattern

discovery (between rows of data or between columns), is whether genes are required to co-express across arrays or within them. Recognizing this allows not only the discovery of cross-clustering arrays (tissues) but also of cross-clustering genes. Employing both types of pattern discovery could be very useful for identifying problem samples and increasing the accuracy of classification techniques.

Much of the information in a set of patterns is ignored by most methods of microarray analysis. For instance, Genes@Work pares down all the discovered patterns to a few, most-interesting ones and only uses these for classification. A similar approach was used in this thesis, where only the top 5, 10 or 20 patterns were considered interesting. This is a straightforward and easy approach but it ignores all the potential information contained in the remaining patterns. For instance, the longest support-three pattern could be used to establish the shortest support-two pattern that would be considered interesting, or a higher support pattern could be used to cluster together multiple lower support patterns. For example, if an interesting pattern existed between streams A and B and another between streams C and D, then an interesting pattern that is supported by all four streams could group A, B, C and D together into a single cluster. In essence lower support patterns are used to reinforce the clusters made by higher support patterns.

In this work, pattern discovery was successfully used on two distinct problems with two different data sets. It was shown that pattern discovery could find biologically meaningful groups of genes in yeast cell-cycle data and that it could reasonably distinguish between two subtypes of lymphoma. Unfortunately, Cluster found similar (occasionally better) results than pattern discovery in yeast cell-cycle data and not all of the selected patterns from the lymphoma data were perfect discriminators between the subtypes. Despite its short-comings, however, pattern discovery was shown to be an effective tool for gene microarray analysis.

# 6 References


Akasaka, T. et al. (2003). "BCL6 genes translocation in follicular lymphoma: a harbinger of eventual transformation to diffuse aggressive lymphoma." Blood 102(4): 1443-1448.

Alizadeh, A., et al. (2000). "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling." Nature 403:503-511.

Barrans, S. et al. (2004). "FOXP identifies a distinct subset of diffuse large B-cell lymphoma (DLBCL) patients with poor outcome." Blood 104(9): 2933-2935.

Baldi P., et al. (2002). DNA Microarrays and Gene Expression. Cambridge University Press.

Brazma, A., et al. (1998). "Approaches to the Automatic Discovery of Patterns in Biosequences." Journal of Computational Biology 5(2): 279-305.

Brazma, A., et al. (2000). "Gene expression data analysis." FEBS Letters 480:17-24.

Bryce, N., et al. (2003). "RAM: A Conserved Signaling Network That Regulates Ace2p Transcriptional Activity and Polarized Morphogenesis." Molecular Biology of the Cell. 14 (9): 3782–3803.

Califano, A., et al. (1999). "SPLASH: Structural Pattern Localization and Analysis by Sequential Histograms." Bioinformatics 16:341-357.

Califano, A., et al. (2000). "Analysis of gene expression microarrays for phenotype classification." Proceedings of the Annual Intelligent Systems in Molecular Biology (ISMB) 2000 8:75-85.

CYGD. 14 March, 2004. Munich Information Centre for Protein Sequences. <http://mips.gsf.de/genre/proj/yeast/index.jsp>.

Draghici, S. (2003). Data Analysis Tools for DNA Microarrays. Chapman & Hall/CRC.

Eisen M., et al. (1998). "Cluster Analysis and Display of Genome-Wide Expression Patterns." Proceedings of the National Academy of Science U.S.A. 95:14863-8.

Faure, R., et al. (1995). "Arrest at the G2/M transition of the cell cycle by protein-tyrosine phosphatase inhibition: studies on a neuronal and a glial cell line." Journal of Cell Biochemistry. 59(3):389-401.

Flick, K., et al. (1998). "Regulation of Cell Size by Glucose Is Exerted via Repression of the CLN1 Promoter." Molecular and Cellular Biology 18(5): 2492-2501.

Floratos, A., et al. (1998). "On The Time Complexity Of The Teiresias Algorithm." IBM Research Report RC 21161.

Gasch, A., et al. (2001). "Genomic expression programs in the response of yeast cells to environmental changes." Molecular Biology of the Cell 11(12):4241-57.

Gascoyne, RD. (2004). "Emerging prognostic factors in diffuse large B cell lymphoma." Current Opinion in Oncology 16(5):436-41.

GO. March 31, 2003. Saccharomyces Genome Databse. <http://genome-www.stanford.edu/Saccharomyces/>.

Hartigan, J. (1975). Clustering Algorithms. John Wiley & Sons. 84-85.

Hans, C. et al. (2004). "Confirmation of the molecular classification of diffuse large B-cell lymphoma by immunochemistry using a tissue microarray." Blood 103(1): 275-282.

Horak, C., et al. (2002). "Complex transcriptional circuitry at the G1/S transition in Saccharomyces cerevisiae." Genes & Develpoment 16(23):3017–3033.
Jaffe, E. S. (2001). Pathology and Genetics of Tumours of the Haemopoeitic and Lymphoid Tissues. IARC WHO Classification of Tumours.
Kluger, Y., et al. (2003). "Spectral biclustering of microarray data: coclustering genes and conditions." Genome Research 13(4):703-16.
Kuehl, R. (1994). Statistical Principle of Research Design and Analysis. Duxbury Press.
LLMPP. January, 2002. Lymphoma/Leukemia Molecular Profiling Project. <http://llmpp.nih.gov/lymphoma/>.
Michele, D. (1997). "Vanadates form insoluble complexes with histones." Biochemie 79(7):457-62.
Pelleg, D. et al. April 3rd, 2003a. Xmeans: Extending K-means with Efficent Estimation of the Number of Clusters. <http://www.autonlab.org/autonweb/documents/papers/pllg:xmns.ps>.
Pelleg, D. April 3rd, 2003b. <http://www-2.cs.cmu.edu/~dpelleg/kmeans.html>.
Rigoutsos, I., et al. (1998). "Combinatorial Pattern Discovery In Biological Sequences: The TEIRESIAS Algorithm." Bioinformatics 14:55-67.
Rigoutsos, I., et al. (2000). "The Emergence of Pattern Discovery Techniques in Computational Biology." Metabolic Engineering. 2(3):159-177.
Rogers, W. et al., (2002). "Analysis of Serine Proteases by Pattern Discovery Using the Tupleware Algorithm." Proceedings of the Annual Intelligent Systems in Molecular Biology (ISMB) 2002 Poster 189B.
Rosenwald, A., et al. (2002). "The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma." New England Journal of Medicine 346(25):1937-47.
Shen, Y. et al (2004). "BCL2 protein expression parallels its mRNA level in normal and malignant B cells." Blood 104(9) 2936-2939.
Simon, I., et al. (2001). "Serial regulation of transcriptional regulators in the yeast cell cycle." Cell 106:697-708.
Spellman P., et al. (1998). "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization." Molecular Biology of the Cell 9: 3273-97.
Stolovitzky, G. et al. "Genes@Work user's guide." March 28th 2003. <http://www.research.ibm.com/FunGen/NewFiles/FGGenesAtWorkDoc.html>.
Suzuki, H. (2002) "A genomic screen for genes upregulated by demethylation and histone deacetylase inhibition in human colorectal cancer." Nature Genetics 31:141-149.
Wasserman, L. "Bayesian Model Selection and Model Averaging." April 3rd, 2003. <http://www.stat.cmu.edu/www/cmu-stats/tr/tr666/tr666.html>.
Wright, G. et al. (2003) "A gene expression-based method to diagnose clinically distinct subgroups of diffuse large B cell lymphoma." PNAS 100:9991-9996.
Zurada, J. (1992). Introduction to Artificial Neural Systems. West Publishing Company.

# Appendix A

**Table A.1. Top 20 highest support patterns with with Xmeans binning $K_{max}$=10**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 35 | 1 | 0.657 | 0, 3, 4, 8, 10, 13, 16, 17, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, | JNK3, |
| 2 | 34 | 1 | 0.382 | 0, 1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 25, 26, 28, 29, 33, 34, 38, 39, 40, 41, 44, 45, 46, | PTK, |
| 3 | 32 | 1 | 0.625 | 1, 2, 3, 4, 8, 9, 11, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, | Hs.203866, |
| 4 | 31 | 1 | 0.516 | 0, 1, 2, 3, 6, 7, 8, 9, 10, 13, 14, 15, 18, 19, 22, 28, 29, 30, 31, 32, 34, 35, 37, 39, 40, 41, 42, 43, 44, 45, 46, | Hs.75859, |
| 5 | 26 | 2 | 0.769 | 3, 4, 8, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, | JNK3, Hs.203866, |
| 6 | 26 | 1 | 0.846 | 9, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, | Hs.120716, |
| 7 | 26 | 1 | 0.615 | 2, 4, 8, 11, 12, 13, 14, 21, 22, 23, 24, 25, 26, 27, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, | Hs.97530, |
| 8 | 26 | 1 | 0.807 | 0, 6, 21, 22, 23, 24, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, | MYBL1, |
| 9 | 25 | 1 | 0.52 | 1, 2, 4, 7, 11, 14, 15, 17, 19, 21, 22, 23, 24, 25, 26, 27, 29, 30, 33, 35, 36, 38, 39, 42, 45, | RAD50, |
| 10 | 25 | 2 | 0.88 | 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, | JNK3, Hs.120716, |
| 11 | 25 | 1 | 0.72 | 3, 5, 6, 7, 9, 10, 23, 24, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 43, 44, 45, 46, | CD21, |
| 12 | 24 | 1 | 0.375 | 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 18, 19, 20, 22, 25, 29, 30, 31, 40, 41, 43, 45, 46, | ID2H, |
| 13 | 24 | 2 | 0.875 | 0, 22, 23, 24, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, | JNK3, MYBL1, |
| 14 | 24 | 1 | 0.375 | 0, 1, 2, 3, 4, 7, 9, 10, 11, 12, 14, 20, 21, 22, 23, 24, 27, 28, 30, 31, 33, 38, 40, 41, | Hs.222808, |
| 15 | 24 | 1 | 0.25 | 0, 1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 18, 19, 20, 21, 23, 24, 27, 29, 32, 38, 40, | DCTD, |
| 16 | 23 | 2 | 0.826 | 9, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 44, | Hs.203866, Hs.120716, |
| 17 | 23 | 1 | 0.826 | 0, 3, 4, 9, 24, 25, 26, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, | KIAA1039, |
| 18 | 23 | 2 | 0.695 | 0, 3, 8, 10, 13, 18, 22, 28, 29, 30, 31, 32, 34, 35, 37, 39, 40, 41, 42, 43, 44, 45, 46, | JNK3, Hs.75859, |
| 19 | 23 | 2 | 0.565 | 0, 3, 8, 10, 13, 16, 17, 18, 22, 23, 25, 26, 28, 29, 33, 34, 38, 39, 40, 41, 44, 45, 46, | JNK3, PTK, |
| 20 | 23 | 2 | 0.391 | 0, 1, 2, 3, 7, 8, 9, 10, 13, 14, 15, 18, 19, 22, 28, 29, 34, 39, 40, 41, 44, 45, 46, | Hs.75859, PTK, |

**Table A.2. Top 20 highest support patterns with with Xmeans binning $K_{max}$=20**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 34 | 1 | 0.382 | 0, 1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 25, 26, 28, 29, 33, 34, 38, 39, 40, 41, 44, 45, 46, | PTK, |
| 2 | 32 | 1 | 0.625 | 1, 2, 3, 4, 8, 9, 11, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, | Hs.203866, |
| 3 | 31 | 1 | 0.516 | 0, 1, 2, 3, 6, 7, 8, 9, 10, 13, 14, 15, 18, 19, 22, 28, 29, 30, 31, 32, 34, 35, 37, 39, 40, 41, 42, 43, 44, 45, 46, | Hs.75859, |
| 4 | 26 | 1 | 0.807 | 0, 6, 21, 22, 23, 24, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, | MYBL1, |
| 5 | 26 | 1 | 0.615 | 2, 4, 8, 11, 12, 13, 14, 21, 22, 23, 24, 25, 26, 27, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, | Hs.97530, |
| 6 | 25 | 1 | 0.72 | 3, 5, 6, 7, 9, 10, 23, 24, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 43, 44, 45, 46, | CD21, |
| 7 | 25 | 1 | 0.52 | 1, 2, 4, 7, 11, 14, 15, 17, 19, 21, 22, 23, 24, 25, 26, 27, 29, 30, 33, 35, 36, 38, 39, 42, 45, | RAD50, |
| 8 | 24 | 1 | 0.25 | 0, 1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 18, 19, 20, 21, 23, 24, 27, 29, 32, 38, 40, | DCTD, |
| 9 | 24 | 1 | 0.375 | 0, 1, 2, 3, 4, 7, 9, 10, 11, 12, 14, 20, 21, 22, 23, 24, 27, 28, 30, 31, 33, 38, 40, 41, | Hs.222808, |
| 10 | 24 | 1 | 0.375 | 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 18, 19, 20, 22, 25, 29, 30, 31, 40, 41, 43, 45, 46, | ID2H, |
| 11 | 23 | 2 | 0.391 | 0, 1, 2, 3, 7, 8, 9, 10, 13, 14, 15, 18, 19, 22, 28, 29, 34, 39, 40, 41, 44, 45, 46, | Hs.75859, PTK, |
| 12 | 23 | 1 | 0.826 | 0, 3, 4, 9, 24, 25, 26, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, | KIAA1039, |
| 13 | 23 | 1 | 0.434 | 2, 3, 5, 6, 7, 8, 10, 12, 13, 14, 18, 19, 20, 24, 25, 27, 28, 29, 34, 36, 38, 40, 46, | Clone.704802, |
| 14 | 23 | 1 | 0.782 | 0, 2, 3, 4, 23, 24, 25, 27, 29, 30, 31, 32, 33, 34, 35, 36, 39, 40, 41, 42, 43, 45, 46, | CD10, |
| 15 | 22 | 2 | 0.681 | 2, 4, 8, 11, 21, 22, 23, 24, 25, 26, 27, 29, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, | Hs.97530, Hs.203866, |
| 16 | 22 | 1 | 0.409 | 0, 3, 6, 8, 9, 10, 11, 12, 14, 15, 16, 18, 19, 24, 25, 27, 30, 31, 32, 33, 36, 38, | Hs.124049, |
| 17 | 22 | 1 | 0.59 | 0, 2, 4, 6, 8, 9, 19, 20, 21, 26, 27, 28, 29, 31, 32, 36, 38, 39, 42, 43, 44, 45, | CAM1, |
| 18 | 22 | 1 | 0.772 | 1, 4, 19, 21, 23, 24, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 43, 45, | TDT, |
| 19 | 21 | 1 | 0.333 | 0, 1, 2, 4, 7, 9, 14, 15, 16, 17, 18, 19, 20, 21, 25, 26, 36, 39, 40, 41, 42, | RGS13, |
| 20 | 21 | 2 | 0.857 | 21, 22, 23, 24, 25, 26, 28, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, | Hs.203866, MYBL1, |

**Table A.3. Top 20 longest patterns with Xmeans binning $K_{max}$=10 (gene names ommitted)**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 2 | 76 | 0.5 | 4, 24, |
| 2 | 2 | 74 | 1 | 30, 31, |
| 3 | 2 | 73 | 0.5 | 21, 33, |
| 4 | 2 | 72 | 1 | 33, 34, |
| 5 | 2 | 71 | 0.5 | 21, 41, |
| 6 | 2 | 70 | 1 | 24, 27, |
| 7 | 2 | 70 | 1 | 45, 46, |
| 8 | 2 | 70 | 0 | 2, 21, |
| 9 | 2 | 69 | 1 | 30, 37, |
| 10 | 2 | 69 | 0 | 1, 2, |
| 11 | 2 | 68 | 1 | 27, 31, |
| 12 | 2 | 68 | 1 | 29, 39, |
| 13 | 2 | 68 | 1 | 37, 40, |
| 14 | 2 | 67 | 1 | 34, 37, |
| 15 | 2 | 67 | 1 | 29, 38, |
| 16 | 2 | 67 | 1 | 24, 25, |
| 17 | 2 | 67 | 0 | 2, 11, |
| 18 | 2 | 67 | 1 | 31, 46, |
| 19 | 2 | 67 | 1 | 33, 46, |
| 20 | 2 | 67 | 0 | 7, 11, |

**Table A.4. Top 20 longest patterns with Xmeans binning $K_{max}$=20 (gene names ommitted)**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 2 | 99 | 1 | 30, 31, |
| 2 | 2 | 99 | 1 | 28, 32, |
| 3 | 2 | 99 | 1 | 33, 46, |
| 4 | 2 | 99 | 0.5 | 21, 33, |
| 5 | 2 | 98 | 0.5 | 4, 24, |
| 6 | 2 | 98 | 1 | 31, 45, |
| 7 | 2 | 97 | 0 | 18, 19, |
| 8 | 2 | 95 | 1 | 37, 40, |
| 9 | 2 | 94 | 1 | 30, 37, |
| 10 | 2 | 94 | 0 | 11, 21, |
| 11 | 2 | 94 | 1 | 33, 34, |
| 12 | 2 | 94 | 1 | 36, 39, |
| 13 | 2 | 93 | 1 | 24, 27, |
| 14 | 2 | 93 | 0 | 7, 18, |
| 15 | 2 | 91 | 1 | 31, 37, |
| 16 | 2 | 91 | 0.5 | 21, 41, |
| 17 | 2 | 91 | 1 | 45, 46, |
| 18 | 2 | 90 | 1 | 26, 34, |
| 19 | 2 | 90 | 1 | 29, 39, |
| 20 | 2 | 90 | 1 | 24, 33, |

**Table A.5. Top 20 highest support patterns with Xmeans binning, $K_{max}$=10, with 100% ABC or GCB support**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 18 | 3 | 1 | 24, 25, 26, 29, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, KCNN3, |
| 2 | 18 | 2 | 0 | 0, 1, 2, 3, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21 | KCNN3, APAF, |
| 3 | 17 | 4 | 1 | 24, 25, 26, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, KCNN3, WASPIP, |
| 4 | 16 | 4 | 1 | 24, 25, 29, 31, 32, 33, 34, 35, 36, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, KCNN3, ALU subfamily SB, |
| 5 | 16 | 2 | 0 | 0, 1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 18, 19, 21 | KCNN3, Clone.1357636, |
| 6 | 16 | 3 | 1 | 25, 26, 29, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, Hs.49614, |
| 7 | 15 | 4 | 1 | 24, 25, 26, 29, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43 | Clone.1356323, Clone.1340456, Clone.1333667, KCNN3, |
| 8 | 15 | 4 | 1 | 24, 25, 26, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43 | Clone.1356323, Clone.1340456, Clone.1333667, WASPIP, |
| 9 | 15 | 4 | 1 | 25, 26, 29, 32, 34, 35, 36, 38, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, KCNN3, Hs.49614, |
| 10 | 15 | 5 | 1 | 24, 25, 31, 32, 33, 34, 35, 36, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, KCNN3, WASPIP, ALU subfamily SB, |
| 11 | 15 | 2 | 0 | 1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 18, 19 | KCNN3, WASPIP, |
| 12 | 15 | 4 | 1 | 25, 26, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, Hs.49614, WASPIP, |
| 13 | 14 | 5 | 1 | 24, 25, 26, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43 | Clone.1356323, Clone.1340456, Clone.1333667, KCNN3, WASPIP, |
| 14 | 14 | 4 | 1 | 24, 26, 29, 31, 32, 33, 34, 35, 36, 37, 38, 40, 43, 45 | Hs.27774, Clone.1356323, Clone.1340456, CD22, |
| 15 | 14 | 4 | 1 | 24, 26, 29, 31, 32, 33, 34, 35, 36, 38, 40, 41, 43, 45 | Clone.1356323, Clone.1340456, CD22, KCNN3, |
| 16 | 14 | 4 | 1 | 24, 26, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 43, 45 | Clone.1356323, Clone.1340456, CD22, WASPIP, |
| 17 | 14 | 4 | 1 | 24, 25, 26, 29, 31, 32, 33, 34, 35, 36, 38, 39, 41, 42 | KIAA1037, Clone.1356323, Clone.1340456, KCNN3, |
| 18 | 14 | 4 | 1 | 24, 26, 29, 31, 32, 33, 34, 35, 36, 38, 40, 43, 45, 46 | Hs.27774, Clone.1356323, Clone.1340456, KCNN3, |
| 19 | 14 | 5 | 1 | 25, 26, 32, 34, 35, 36, 38, 39, 40, 41, 42, 43, 45, 46 | Clone.1356323, Clone.1340456, KCNN3, Hs.49614, WASPIP, |
| 20 | 14 | 3 | 0 | 1, 2, 3, 5, 7, 8, 10, 11, 12, 13, 14, 15, 18, 19 | KCNN3, WASPIP, APAF, |

**Table A.6. Top 20 highest support patterns with Xmeans binning, $K_{max}$=20, with 100% ABC or GCB support**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 16 | 2 | 1 | 25, 26, 29, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46 | Hs.19399, ALU subfamily SB, |
| 2 | 15 | 3 | 1 | 24, 25, 26, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43 | Hs.19399, Clone.2005, FMR2, |
| 3 | 15 | 3 | 1 | 25, 26, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46 | Hs.19399, ALU subfamily SB, FMR2, |
| 4 | 14 | 3 | 1 | 24, 26, 29, 31, 32, 33, 34, 35, 36, 37, 38, 40, 43, 45 | CD21, Hs.19399, Hs.123344, |
| 5 | 14 | 3 | 1 | 24, 26, 31, 32, 33, 34, 35, 36, 37, 38, 40, 43, 45, 46 | CD21, Hs.19399, FMR2, |
| 6 | 14 | 3 | 1 | 25, 26, 31, 33, 34, 35, 36, 37, 39, 41, 42, 43, 45, 46 | Hs.19399, FMR2, Hs.47232, |
| 7 | 14 | 3 | 1 | 24, 25, 26, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42 | Hs.193367, Hs.19399, FMR2, |
| 8 | 14 | 3 | 1 | 24, 26, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 43, 45 | Hs.19399, Hs.123344, FMR2, |
| 9 | 13 | 4 | 1 | 24, 26, 31, 32, 33, 34, 35, 36, 37, 38, 40, 43, 45 | CD21, Hs.19399, Hs.123344, FMR2, |
| 10 | 13 | 3 | 1 | 26, 28, 30, 31, 33, 34, 35, 36, 37, 43, 44, 45, 46 | CD21, FMR2, Hs.47232, |
| 11 | 13 | 3 | 1 | 25, 30, 31, 33, 34, 35, 36, 39, 41, 42, 43, 45, 46 | FMR2, Hs.173108, Hs.47232, |
| 12 | 13 | 4 | 1 | 24, 25, 26, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42 | Hs.193367, Hs.19399, Clone.2005, FMR2, |
| 13 | 13 | 3 | 1 | 25, 26, 29, 32, 34, 36, 37, 38, 39, 40, 41, 42, 43 | Hs.19399, Clone.2005, ALU subfamily SB, |
| 14 | 13 | 3 | 1 | 24, 26, 31, 32, 34, 36, 37, 38, 39, 42, 43, 45, 46 | Hs.19399, FMR2, NME4, |
| 15 | 13 | 3 | 1 | 25, 29, 32, 34, 35, 36, 39, 40, 41, 42, 43, 45, 46 | Hs.19399, ALU subfamily SB, Hs.173108, |
| 16 | 13 | 2 | 1 | 25, 26, 29, 34, 35, 36, 37, 39, 42, 43, 44, 45, 46 | Clone.1351325, ALU subfamily SB, |
| 17 | 13 | 1 | 1 | 26, 29, 31, 32, 34, 35, 37, 38, 39, 40, 41, 44, 45 | Hs.120716, |
| 18 | 13 | 2 | 1 | 24, 26, 30, 32, 35, 36, 37, 38, 39, 40, 41, 44, 46 | Clone.1339726, FMR2, |
| 19 | 12 | 4 | 1 | 24, 26, 29, 31, 32, 33, 34, 36, 37, 38, 40, 43 | CD21, Hs.19399, Hs.123344, Clone.2005, |
| 20 | 12 | 3 | 1 | 26, 29, 32, 34, 35, 36, 37, 38, 40, 43, 45, 46 | CD21, Hs.19399, ALU subfamily SB, |

**Table A.7. Top 20 highest support patterns with Xmeans binning, $K_{max}$=10 and $G_{min}$=8**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 10 | 8 | 1 | 32, 34, 35, 36, 38, 39, 41, 42, 43, 45, | KIAA1039, JNK3, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, |
| 2 | 9 | 8 | 0.777 | 22, 23, 32, 34, 36, 38, 39, 41, 42, | Hs.97530, JNK3, Hs.203866, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, |
| 3 | 9 | 8 | 0.888 | 22, 32, 34, 36, 38, 39, 41, 42, 43, | JNK3, Hs.203866, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, |
| 4 | 9 | 8 | 0.888 | 23, 28, 32, 34, 36, 39, 41, 42, 43, | JNK3, Hs.203866, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 5 | 9 | 8 | 0.888 | 23, 32, 34, 36, 39, 40, 41, 42, 43, | JNK3, Hs.203866, Hs.120716, Hs.224323, FMR2, MYBL1, CD10, Hs.106771, |
| 6 | 9 | 8 | 0.888 | 22, 32, 34, 35, 39, 41, 42, 43, 45, | JNK3, Hs.75859, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, |
| 7 | 9 | 9 | 1 | 32, 34, 35, 36, 39, 41, 42, 43, 45, | KIAA1039, JNK3, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, CD10, |
| 8 | 9 | 8 | 0.888 | 22, 32, 34, 35, 36, 38, 39, 41, 42, | Hs.97530, JNK3, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, |
| 9 | 9 | 8 | 0.888 | 22, 34, 35, 36, 39, 41, 42, 43, 45, | JNK3, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, DCTD, |
| 10 | 9 | 8 | 1 | 32, 34, 35, 39, 40, 41, 42, 43, 45, | KIAA1039, JNK3, Hs.75859, Hs.120716, Hs.224323, FMR2, MYBL1, CD10, |
| 11 | 9 | 8 | 0.888 | 22, 28, 34, 35, 39, 41, 42, 43, 45, | JNK3, Hs.75859, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, DCTD, |
| 12 | 9 | 8 | 1 | 32, 34, 35, 36, 39, 40, 41, 42, 43, | KIAA1039, JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, CD10, Hs.106771, |
| 13 | 9 | 8 | 0.888 | 23, 32, 34, 35, 36, 39, 41, 42, 43, | JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, CD10, Hs.106771, |
| 14 | 9 | 8 | 0.888 | 22, 28, 34, 37, 39, 41, 42, 43, 45, | JNK3, Hs.75859, Hs.124049, Hs.224323, FMR2, MYBL1, MYBL1, DCTD, |
| 15 | 9 | 8 | 1 | 25, 34, 35, 36, 39, 42, 43, 45, 46, | KIAA1039, JNK3, Hs.222808, Hs.120716, Hs.224323, MYBL1, CD10, DCTD, |
| 16 | 8 | 8 | 0.875 | 23, 28, 32, 34, 35, 36, 37, 43, | CD21, CD21, JNK3, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 17 | 8 | 8 | 0.75 | 22, 23, 34, 35, 36, 38, 39, 41, | Hs.97530, JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, Clone.1334486, |
| 18 | 8 | 8 | 0.75 | 22, 23, 34, 36, 37, 38, 39, 41, | Hs.97530, JNK3, Hs.203866, Hs.224323, FMR2, MYBL1, MYBL1, Clone.1334486, |
| 19 | 8 | 8 | 0.875 | 22, 28, 32, 40, 41, 42, 43, 44, | JNK3, Clone.1333519, Hs.75859, Hs.203866, Hs.120716, Hs.224323, Clone.825199, MYBL1, |
| 20 | 8 | 8 | 1 | 29, 32, 34, 39, 40, 41, 42, 43, | KIAA1039, JNK3, Hs.75859, Hs.203866, Hs.120716, Hs.224323, CD10, Hs.106771, |

**Table A.8. Top 20 highest support patterns with Xmeans binning, $K_{max}$=20 and $G_{min}$=8**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 8 | 8 | 1 | 28, 34, 35, 37, 39, 41, 42, 43, | Hs.75859, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, DCTD, DCTD, |
| 2 | 8 | 8 | 1 | 34, 35, 36, 39, 41, 42, 43, 45, | KIAA1039, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, CD10, DCTD, |
| 3 | 8 | 8 | 0.875 | 22, 28, 34, 37, 39, 41, 42, 43, | Hs.75859, Hs.203866, Hs.124049, Hs.224323, FMR2, MYBL1, MYBL1, DCTD, |
| 4 | 8 | 8 | 1 | 32, 34, 35, 39, 41, 42, 43, 45, | KIAA1039, Hs.75859, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, CD10, |
| 5 | 8 | 8 | 1 | 32, 34, 35, 36, 39, 41, 42, 43, | KIAA1039, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, CD10, Hs.106771, |
| 6 | 7 | 8 | 1 | 30, 35, 36, 37, 39, 41, 44, | ALOX, Hs.125815, PI3K, MYBL1, MYBL1, Clone.1334486, BCL2, DCTD, |
| 7 | 7 | 8 | 0.714 | 9, 13, 35, 37, 40, 41, 44, | ALOX, Hs.125815, PI3K, Hs.75859, Hs.24724, Clone.1234554, Clone.825920, Clone.1288046, |
| 8 | 7 | 9 | 1 | 32, 34, 35, 36, 37, 40, 43, | CD21, Hs.125815, KIAA1039, Hs.28355, TDT, Hs.224323, FMR2, MYBL1, Hs.106771, |
| 9 | 7 | 8 | 1 | 32, 34, 35, 37, 39, 40, 43, | Hs.125815, KIAA1039, Hs.28355, Hs.75859, Hs.224323, FMR2, MYBL1, Hs.106771, |
| 10 | 7 | 8 | 1 | 32, 34, 36, 37, 39, 40, 43, | Hs.125815, KIAA1039, Hs.28355, Hs.203866, Hs.224323, FMR2, MYBL1, Hs.106771, |
| 11 | 7 | 8 | 1 | 32, 34, 35, 36, 37, 39, 43, | Hs.125815, KIAA1039, Hs.28355, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 12 | 7 | 8 | 1 | 32, 34, 35, 36, 39, 40, 43, | Hs.125815, KIAA1039, Hs.28355, Hs.224323, FMR2, MYBL1, CD10, Hs.106771, |
| 13 | 7 | 8 | 1 | 32, 35, 36, 37, 39, 40, 43, | Hs.125815, KIAA1039, Hs.28355, Hs.224323, FMR2, MYBL1, KIAA0870, Hs.106771, |
| 14 | 7 | 8 | 1 | 32, 34, 35, 36, 37, 39, 42, | Hs.97530, KIAA1039, Hs.28355, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 15 | 7 | 8 | 1 | 32, 34, 37, 39, 40, 42, 43, | KIAA1039, Hs.28355, Hs.75859, Hs.203866, Hs.224323, FMR2, MYBL1, Hs.106771, |
| 16 | 7 | 8 | 1 | 32, 34, 35, 37, 39, 42, 43, | KIAA1039, Hs.28355, Hs.75859, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 17 | 7 | 8 | 1 | 32, 34, 35, 39, 40, 42, 43, | KIAA1039, Hs.28355, Hs.75859, Hs.224323, FMR2, MYBL1, CD10, Hs.106771, |
| 18 | 7 | 8 | 1 | 32, 35, 37, 39, 40, 42, 43, | KIAA1039, Hs.28355, Hs.75859, Hs.224323, FMR2, MYBL1, KIAA0870, Hs.106771, |
| 19 | 7 | 9 | 1 | 32, 34, 36, 37, 39, 42, 43, | KIAA1039, Hs.28355, Hs.203866, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, ETV6, |
| 20 | 7 | 8 | 1 | 32, 34, 36, 39, 40, 42, 43, | KIAA1039, Hs.28355, Hs.203866, Hs.224323, FMR2, MYBL1, CD10, Hs.106771, |

**Table A.9. Top 20 highest support patterns with Xmeans binning, $K_{max}$=10, ABC<50% and $G_{min}$=8**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 6 | 8 | 0 | 2, 10, 11, 13, 14, 19, | KIAA1039, Hs.28355, Hs.120716, MYBL1, DCTD, TRKC, ID2H, PTK, |
| 2 | 6 | 8 | 0.333 | 2, 21, 22, 23, 29, 39, | RAD50, Hs.97530, Clone.1339105, Hs.203866, GCAM1, Clone.1334486, Hs.226955, PTK, |
| 3 | 6 | 8 | 0.333 | 4, 11, 21, 23, 24, 38, | RAD50, Hs.97530, Clone.1333841, MYOIC, Hs.75859, Hs.203866, Hs.222808, DCTD, |
| 4 | 6 | 8 | 0.333 | 2, 7, 11, 21, 29, 33, | RAD50, ALOX, CD27, CD27, CD27, Hs.226955, Clone.682692, PTK, |
| 5 | 6 | 8 | 0 | 5, 10, 13, 14, 18, 19, | Clone.704802, Hs.120716, MYBL1, CD27, BRCA2, DCTD, ID2H, PTK, |
| 6 | 6 | 8 | 0 | 7, 10, 13, 14, 18, 19, | Clone.704802, Hs.75859, Hs.120716, MYBL1, MYBL1, BRCA2, ID2H, PTK, |
| 7 | 6 | 8 | 0 | 2, 7, 13, 14, 18, 19, | JMJD1B, WASPIP, Clone.704802, Hs.75859, Hs.120716, MYBL1, ID2H, PTK, |
| 8 | 6 | 8 | 0 | 2, 7, 10, 13, 14, 19, | KIAA1039, Hs.28355, Clone.704802, Hs.75859, Hs.120716, MYBL1, ID2H, PTK, |
| 9 | 6 | 10 | 0 | 7, 8, 13, 14, 15, 18, | WASPIP, Hs.75859, Hs.120716, MYBL1, MYBL1, OGG1, ALU subfamily J, ALU subfamily J, BRCA2, PTK, |
| 10 | 6 | 8 | 0.333 | 1, 3, 4, 23, 24, 31, | FAK, Hs.203866, Hs.222808, Hs.161905, CD10, RGS13, TCEB3, Clone.2015, |
| 11 | 6 | 8 | 0.333 | 8, 9, 19, 21, 38, 39, | Hs.19399, WASPIP, WASPIP, Hs.203866, GCAM1, CAM1, PTK, ALU subfamily C, |
| 12 | 6 | 8 | 0.333 | 2, 8, 19, 21, 29, 38, | ALOX, WASPIP, Hs.203866, GCAM1, CAM1, DCTD, PTK, ALU subfamily C, |
| 13 | 6 | 8 | 0 | 2, 7, 8, 13, 14, 19, | KIAA1039, Hs.28355, WASPIP, Clone.704802, Hs.75859, Hs.120716, MYBL1, PTK, |
| 14 | 6 | 8 | 0 | 7, 8, 13, 14, 18, 19, | WASPIP, Clone.704802, Hs.75859, Hs.120716, MYBL1, MYBL1, BRCA2, PTK, |
| 15 | 6 | 8 | 0 | 8, 13, 14, 15, 18, 19, | WASPIP, Hs.75859, Hs.120716, MYBL1, MYBL1, Hs.161905, BRCA2, PTK, |
| 16 | 6 | 9 | 0 | 7, 8, 10, 13, 14, 18, | Clone.704802, TDT, Hs.75859, Hs.120716, MYBL1, MYBL1, OGG1, BRCA2, PTK, |
| 17 | 6 | 8 | 0 | 3, 8, 10, 14, 18, 19, | Clone.704802, Hs.75859, Hs.124049, Hs.120716, MYBL1, MYBL1, DCTD, PTK, |
| 18 | 6 | 8 | 0 | 2, 7, 8, 10, 13, 14, | CSNK1G2, KIAA1039, Hs.28355, Clone.704802, Hs.75859, Hs.120716, MYBL1, PTK, |
| 19 | 6 | 9 | 0 | 7, 8, 10, 13, 14, 19, | KIAA1039, Hs.28355, Clone.704802, Hs.75859, Hs.120716, MYBL1, MYBL1, BRCA2, PTK, |
| 20 | 6 | 8 | 0 | 2, 8, 10, 13, 14, 19, | KIAA1039, Hs.28355, Clone.704802, Hs.75859, Hs.120716, MYBL1, DCTD, PTK, |

**Table A.10. Top 20 highest support patterns with Xmeans binning, $K_{max}$=20, ABC<50% and $G_{min}$=8**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 6 | 8 | 0 | 7, 8, 10, 13, 14, 18, | Clone.704802, TDT, Hs.75859, MYBL1, MYBL1, OGG1, BRCA2, PTK, |
| 2 | 6 | 8 | 0 | 7, 8, 13, 14, 15, 18, | WASPIP, Hs.75859, MYBL1, MYBL1, OGG1, ALU subfamily J, BRCA2, PTK, |
| 3 | 6 | 8 | 0 | 2, 8, 11, 14, 21, 23, | Hs.97530, KIAA1039, Hs.28355, WASPIP, JAW1, POU4F1, DCTD, PTK, |
| 4 | 6 | 8 | 0 | 7, 8, 10, 13, 14, 19, | KIAA1039, Hs.28355, Clone.704802, Hs.75859, MYBL1, MYBL1, BRCA2, PTK, |
| 5 | 6 | 8 | 0.166 | 2, 13, 14, 18, 19, 29, | WASPIP, Clone.704802, Hs.75859, MYBL1, Hs.46913, DCTD, ID2H, PTK, |
| 6 | 6 | 8 | 0.333 | 2, 8, 19, 21, 29, 38, | ALOX, WASPIP, Hs.203866, GCAM1, CAM1, DCTD, PTK, ALU subfamily C, |
| 7 | 6 | 8 | 0.333 | 4, 11, 21, 23, 24, 38, | RAD50, Hs.97530, Clone.1333841, MYOIC, Hs.75859, Hs.203866, Hs.222808, DCTD, |
| 8 | 6 | 8 | 0.333 | 8, 9, 19, 21, 38, 39, | Hs.19399, WASPIP, WASPIP, Hs.203866, GCAM1, CAM1, PTK, ALU subfamily C, |
| 9 | 6 | 8 | 0.333 | 2, 21, 22, 23, 29, 39, | RAD50, Hs.97530, Clone.1339105, Hs.203866, GCAM1, Clone.1334486, Hs.226955, PTK, |
| 10 | 6 | 8 | 0.333 | 2, 7, 11, 21, 29, 33, | RAD50, ALOX, CD27, CD27, CD27, Hs.226955, Clone.682692, PTK, |
| 11 | 6 | 8 | 0 | 2, 7, 13, 14, 18, 19, | JMJD1B, WASPIP, Clone.704802, Hs.75859, MYBL1, Hs.46913, ID2H, PTK, |
| 12 | 6 | 8 | 0.166 | 7, 13, 14, 18, 19, 29, | WASPIP, Clone.704802, Hs.75859, MYBL1, BRCA2, Hs.46913, ID2H, PTK, |
| 13 | 6 | 8 | 0.333 | 11, 18, 19, 21, 39, 41, | Clone.1356654, Hs.62684, WASPIP, BCL7A, OP1, CD44, FLICE, PTK, |
| 14 | 6 | 8 | 0.333 | 11, 18, 19, 21, 25, 41, | Hs.62684, WASPIP, WASPIP, BCL7A, OP1, IRF4, FLICE, PTK, |
| 15 | 5 | 9 | 0.4 | 4, 11, 21, 25, 34, | KIAA1037, Hs.97530, ALOX, Clone.1333841, MYOIC, KIAA0808, Hs.203866, IL4R, GENE15X__Unknown__Clone_2013_, |
| 16 | 5 | 9 | 0.4 | 8, 11, 21, 25, 41, | KIAA1037, Hs.97530, WASPIP, WASPIP, JAW1, Hs.203866, OP1, FLICE, PTK, |
| 17 | 5 | 9 | 0.4 | 4, 11, 21, 25, 41, | KIAA1037, Hs.97530, JAW1, KIAA0808, Hs.203866, BCL7A, IL4R, FLICE, GENE15X__Unknown__Clone_2013_, |
| 18 | 5 | 8 | 0.2 | 7, 8, 11, 20, 25, | KIAA1037, ALOX, WASPIP, WASPIP, ALU subfamily SB, JAW1, TDT, Hs.192708 (highly sim. to mybA), |
| 19 | 5 | 8 | 0.2 | 1, 11, 18, 21, 41, | KIAA1037, Clone.1356654, Hs.62684, BCL7A, IL4R, IRF4, GENE15X__Unknown__Clone_2013_, PTK, |
| 20 | 5 | 8 | 0.2 | 7, 11, 18, 21, 41, | KIAA1037, Clone.1356654, WASPIP, WASPIP, OP1, IL4R, IRF4, PTK, |

**Table A.11. Top 20 longest patterns with Xmeans binning, $K_{max}$=10, $S_{min}$=5 (gene names ommitted)**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 5 | 16 | 1 | 34, 37, 39, 40, 41, |
| 2 | 5 | 16 | 1 | 34, 36, 39, 42, 43, |
| 3 | 5 | 16 | 1 | 32, 34, 36, 39, 43, |
| 4 | 5 | 16 | 1 | 34, 35, 39, 42, 43, |
| 5 | 5 | 15 | 1 | 32, 34, 39, 40, 41, |
| 6 | 5 | 15 | 1 | 31, 33, 36, 45, 46, |
| 7 | 5 | 15 | 1 | 34, 37, 39, 41, 42, |
| 8 | 5 | 15 | 1 | 34, 39, 41, 42, 45, |
| 9 | 5 | 15 | 1 | 34, 36, 37, 39, 42, |
| 10 | 5 | 15 | 0.6 | 11, 21, 24, 33, 38, |
| 11 | 5 | 15 | 1 | 32, 35, 36, 37, 43, |
| 12 | 5 | 15 | 1 | 32, 34, 35, 36, 43, |
| 13 | 5 | 15 | 1 | 34, 37, 39, 42, 43, |
| 14 | 5 | 15 | 1 | 34, 39, 41, 42, 43, |
| 15 | 5 | 15 | 1 | 32, 36, 39, 42, 43, |
| 16 | 5 | 15 | 1 | 32, 34, 35, 39, 41, |
| 17 | 5 | 15 | 1 | 34, 35, 39, 41, 42, |
| 18 | 5 | 15 | 1 | 32, 35, 36, 39, 41, |
| 19 | 5 | 15 | 1 | 32, 35, 36, 37, 40, |
| 20 | 5 | 15 | 1 | 32, 35, 36, 37, 39, |

**Table A.12. Top 20 longest patterns with Xmeans binning, $K_{max}$=20, $S_{min}$=5 (gene names ommitted)**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 5 | 16 | 1 | 34, 37, 39, 40, 41, |
| 2 | 5 | 15 | 1 | 34, 35, 36, 37, 43, |
| 3 | 5 | 15 | 0.6 | 11, 21, 24, 33, 38, |
| 4 | 5 | 14 | 1 | 30, 35, 36, 37, 44, |
| 5 | 5 | 14 | 1 | 32, 35, 36, 37, 39, |
| 6 | 5 | 14 | 1 | 32, 34, 36, 39, 43, |
| 7 | 5 | 14 | 1 | 32, 35, 36, 37, 40, |
| 8 | 5 | 14 | 1 | 34, 36, 37, 39, 42, |
| 9 | 5 | 14 | 1 | 34, 36, 39, 42, 43, |
| 10 | 5 | 14 | 1 | 34, 37, 39, 42, 43, |
| 11 | 5 | 14 | 1 | 34, 35, 39, 42, 43, |
| 12 | 5 | 14 | 1 | 34, 37, 39, 41, 42, |
| 13 | 5 | 14 | 1 | 34, 36, 37, 39, 41, |
| 14 | 5 | 14 | 0.6 | 11, 21, 24, 27, 33, |
| 15 | 5 | 13 | 1 | 24, 27, 30, 31, 38, |
| 16 | 5 | 13 | 1 | 34, 35, 36, 42, 43, |
| 17 | 5 | 13 | 1 | 31, 33, 36, 45, 46, |
| 18 | 5 | 13 | 0.4 | 1, 3, 4, 24, 27, |
| 19 | 5 | 13 | 1 | 35, 37, 40, 41, 44, |
| 20 | 5 | 13 | 1 | 37, 39, 40, 41, 45, |

**Table A.13. Top 20 longest patterns with Xmeans binning, $K_{max}$=10, $S_{min}$=5, ABC<50% (gene names ommitted)**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 5 | 12 | 0.4 | 1, 3, 4, 24, 31, |
| 2 | 5 | 12 | 0.4 | 1, 3, 4, 24, 27, |
| 3 | 5 | 12 | 0.4 | 0, 3, 4, 24, 31, |
| 4 | 5 | 12 | 0.2 | 2, 7, 11, 21, 33, |
| 5 | 5 | 12 | 0.4 | 7, 11, 21, 33, 38, |
| 6 | 5 | 12 | 0 | 7, 8, 13, 14, 18, |
| 7 | 5 | 12 | 0.4 | 3, 4, 23, 24, 27, |
| 8 | 5 | 12 | 0.4 | 3, 4, 23, 24, 31, |
| 9 | 5 | 12 | 0 | 7, 8, 10, 13, 14, |
| 10 | 5 | 11 | 0 | 2, 11, 13, 14, 19, |
| 11 | 5 | 11 | 0.4 | 21, 22, 23, 30, 38, |
| 12 | 5 | 11 | 0.2 | 1, 2, 7, 21, 27, |
| 13 | 5 | 11 | 0.4 | 2, 4, 21, 29, 38, |
| 14 | 5 | 11 | 0.4 | 4, 21, 23, 27, 38, |
| 15 | 5 | 11 | 0.4 | 2, 11, 21, 29, 38, |
| 16 | 5 | 11 | 0.4 | 11, 21, 23, 24, 38, |
| 17 | 5 | 11 | 0.4 | 11, 21, 22, 33, 38, |
| 18 | 5 | 11 | 0.4 | 11, 21, 22, 26, 33, |
| 19 | 5 | 11 | 0.4 | 11, 21, 23, 24, 33, |
| 20 | 5 | 11 | 0.4 | 11, 21, 23, 24, 27, |

**Table A.14. Top 20 longest patterns with Xmeans binning, $K_{max}$=20, $S_{min}$=5, ABC<50%**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 5 | 13 | 0.4 | 1, 3, 4, 24, 27, |
| 2 | 5 | 13 | 0.4 | 1, 3, 4, 24, 31, |
| 3 | 5 | 12 | 0.2 | 2, 7, 11, 21, 33, |
| 4 | 5 | 11 | 0.4 | 11, 18, 21, 25, 41, |
| 5 | 5 | 11 | 0.2 | 1, 2, 7, 21, 27, |
| 6 | 5 | 11 | 0 | 7, 8, 10, 13, 14, |
| 7 | 5 | 11 | 0 | 7, 13, 14, 18, 19, |
| 8 | 5 | 11 | 0.4 | 1, 3, 4, 27, 31, |
| 9 | 5 | 11 | 0.4 | 11, 21, 22, 26, 33, |
| 10 | 5 | 11 | 0.4 | 3, 4, 23, 24, 27, |
| 11 | 5 | 11 | 0.4 | 4, 21, 23, 27, 38, |
| 12 | 5 | 11 | 0 | 2, 8, 11, 21, 23, |
| 13 | 5 | 11 | 0.4 | 11, 21, 23, 24, 38, |
| 14 | 5 | 11 | 0.4 | 2, 11, 21, 29, 38, |
| 15 | 5 | 11 | 0.4 | 2, 4, 21, 29, 38, |
| 16 | 5 | 11 | 0.4 | 21, 22, 23, 30, 38, |
| 17 | 5 | 11 | 0.4 | 2, 11, 21, 24, 33, |
| 18 | 5 | 11 | 0.4 | 7, 11, 21, 33, 38, |
| 19 | 5 | 10 | 0.4 | 4, 11, 23, 24, 38, |
| 20 | 5 | 10 | 0.4 | 11, 19, 21, 25, 38, |

**Table A.15. Top 20 longest patterns with Xmeans binning, $K_{max}$=10, $S_{min}$=10**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 10 | 8 | 1 | 32, 34, 35, 36, 38, 39, 41, 42, 43, 45, | KIAA1039, JNK3, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, |
| 2 | 10 | 7 | 1 | 24, 26, 31, 32, 33, 34, 36, 38, 40, 43, | CD21, KIAA1039, JNK3, TDT, Hs.203866, Hs.120716, MYBL1, |
| 3 | 10 | 7 | 0.9 | 23, 24, 30, 31, 32, 33, 34, 36, 40, 43, | CD21, JNK3, TDT, Hs.203866, Hs.120716, MYBL1, CD10, |
| 4 | 10 | 7 | 1 | 24, 25, 31, 32, 33, 34, 36, 39, 41, 42, | Hs.97530, KIAA1039, JNK3, Hs.203866, Hs.120716, MYBL1, CD10, |
| 5 | 10 | 7 | 0.8 | 22, 23, 32, 34, 36, 37, 38, 39, 41, 42, | Hs.97530, JNK3, Hs.203866, Hs.224323, FMR2, MYBL1, MYBL1, |
| 6 | 11 | 7 | 0.818 | 22, 23, 28, 32, 34, 36, 38, 39, 41, 42, 43, | JNK3, Hs.203866, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, |
| 7 | 10 | 7 | 0.9 | 23, 28, 32, 34, 36, 39, 40, 41, 42, 43, | JNK3, Hs.203866, Hs.120716, Hs.224323, FMR2, MYBL1, Hs.106771, |
| 8 | 10 | 7 | 0.9 | 23, 28, 32, 34, 36, 37, 39, 41, 42, 43, | JNK3, Hs.203866, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 9 | 11 | 7 | 0.909 | 22, 32, 34, 35, 36, 38, 39, 41, 42, 43, 45, | JNK3, Hs.120716, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, |
| 10 | 10 | 7 | 0.9 | 22, 28, 32, 34, 35, 39, 41, 42, 43, 45, | JNK3, Hs.75859, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, |
| 11 | 10 | 7 | 1 | 32, 34, 35, 36, 39, 40, 41, 42, 43, 45, | KIAA1039, JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, CD10, |
| 12 | 10 | 7 | 0.8 | 22, 23, 32, 34, 35, 36, 38, 39, 41, 42, | Hs.97530, JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, |
| 13 | 10 | 7 | 0.9 | 23, 32, 34, 35, 36, 39, 41, 42, 43, 45, | JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, CD10, |
| 14 | 10 | 7 | 0.9 | 23, 28, 32, 34, 35, 36, 39, 41, 42, 43, | JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 15 | 10 | 7 | 0.9 | 22, 28, 34, 35, 36, 39, 41, 42, 43, 45, | JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, MYBL1, DCTD, |
| 16 | 10 | 7 | 0.9 | 23, 32, 34, 35, 36, 39, 40, 41, 42, 43, | JNK3, Hs.120716, Hs.224323, FMR2, MYBL1, CD10, Hs.106771, |
| 17 | 10 | 7 | 0.9 | 22, 28, 34, 35, 37, 39, 41, 42, 43, 45, | JNK3, Hs.75859, Hs.224323, FMR2, MYBL1, MYBL1, DCTD, |
| 18 | 10 | 7 | 1 | 25, 26, 34, 35, 36, 39, 42, 43, 45, 46, | KIAA1039, JNK3, Hs.222808, Hs.120716, Hs.224323, MYBL1, DCTD, |
| 19 | 10 | 7 | 1 | 24, 31, 32, 33, 34, 35, 36, 40, 43, 45, | CD21, KIAA1039, JNK3, TDT, Hs.120716, MYBL1, CD10, |
| 20 | 10 | 7 | 1 | 32, 34, 35, 39, 40, 41, 42, 43, 45, 46, | KIAA1039, JNK3, Hs.75859, Hs.120716, Hs.224323, MYBL1, CD10, |

**Table A.16. Top 20 longest patterns with Xmeans binning, $K_{max}$=20, $S_{min}$=10**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 10 | 6 | 0.9 | 22, 28, 34, 35, 37, 39, 41, 42, 43, 45, | Hs.75859, Hs.224323, FMR2, MYBL1, MYBL1, DCTD, |
| 2 | 10 | 6 | 0.8 | 22, 23, 32, 34, 36, 37, 38, 39, 41, 42, | Hs.97530, Hs.203866, Hs.224323, FMR2, MYBL1, MYBL1, |
| 3 | 10 | 6 | 0.9 | 23, 28, 32, 34, 36, 37, 39, 41, 42, 43, | Hs.203866, Hs.224323, FMR2, MYBL1, MYBL1, Hs.106771, |
| 4 | 10 | 6 | 1 | 32, 34, 35, 36, 38, 39, 41, 42, 43, 45, | KIAA1039, Hs.224323, Hs.192708 (highly sim. to mybA), FMR2, MYBL1, MYBL1, |
| 5 | 10 | 5 | 0.9 | 23, 24, 26, 30, 31, 36, 37, 38, 40, 43, | KIAA1037, CD21, TDT, Hs.203866, MYBL1, |
| 6 | 10 | 5 | 1 | 24, 26, 29, 31, 32, 33, 34, 36, 37, 38, | CD21, Hs.97530, KIAA1039, TDT, Hs.203866, |
| 7 | 10 | 5 | 1 | 24, 26, 31, 32, 33, 34, 35, 36, 37, 38, | CD21, Hs.97530, KIAA1039, TDT, MYBL1, |
| 8 | 10 | 5 | 0.9 | 23, 24, 26, 31, 32, 33, 34, 36, 37, 38, | CD21, Hs.97530, TDT, Hs.203866, MYBL1, |
| 9 | 11 | 5 | 1 | 24, 26, 31, 32, 33, 34, 36, 37, 38, 40, 43, | CD21, KIAA1039, TDT, Hs.203866, MYBL1, |
| 10 | 10 | 5 | 1 | 26, 32, 34, 35, 36, 37, 38, 40, 43, 45, | CD21, KIAA1039, TDT, Hs.224323, MYBL1, |
| 11 | 10 | 5 | 1 | 24, 31, 32, 33, 34, 35, 36, 40, 43, 45, | CD21, KIAA1039, TDT, MYBL1, CD10, |
| 12 | 10 | 5 | 1 | 24, 26, 31, 32, 34, 36, 37, 38, 43, 45, | CD21, KIAA1039, TDT, MYBL1, ETV6, |
| 13 | 10 | 5 | 0.9 | 23, 24, 30, 31, 32, 33, 34, 36, 40, 43, | CD21, TDT, Hs.203866, MYBL1, CD10, |
| 14 | 10 | 5 | 0.9 | 23, 32, 34, 35, 36, 37, 38, 40, 43, 45, | CD21, TDT, Hs.224323, FMR2, MYBL1, |
| 15 | 10 | 5 | 0.9 | 23, 28, 32, 34, 35, 36, 37, 38, 43, 45, | CD21, Hs.224323, FMR2, MYBL1, MYBL1, |
| 16 | 10 | 5 | 1 | 28, 30, 34, 35, 37, 39, 41, 42, 43, 44, | Hs.75859, MYBL1, MYBL1, DCTD, DCTD, |
| 17 | 10 | 5 | 1 | 28, 30, 31, 34, 37, 39, 41, 42, 43, 44, | Hs.75859, Hs.203866, MYBL1, DCTD, DCTD, |
| 18 | 10 | 5 | 0.9 | 22, 26, 34, 35, 36, 37, 39, 41, 44, 46, | Hs.125815, Hs.224323, MYBL1, Clone.1334486, DCTD, |
| 19 | 10 | 5 | 1 | 25, 34, 35, 36, 39, 41, 42, 43, 45, 46, | KIAA1039, Hs.224323, MYBL1, CD10, DCTD, |
| 20 | 11 | 5 | 0.909 | 22, 28, 34, 35, 36, 37, 39, 41, 42, 43, 45, | Hs.224323, FMR2, MYBL1, MYBL1, DCTD, |

**Table A.17. Top 20 longest patterns with Xmeans binning, $K_{max}$=10, $S_{min}$=10, ABC<50%**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 10 | 5 | 0 | 1, 3, 7, 8, 10, 13, 14, 15, 18, 19, | Hs.75859, Hs.120716, MYBL1, MYBL1, PTK, |
| 2 | 10 | 5 | 0 | 2, 3, 5, 8, 10, 12, 13, 14, 18, 19, | Clone.704802, Hs.120716, MYBL1, DCTD, PTK, |
| 3 | 10 | 4 | 0.4 | 2, 4, 11, 21, 22, 23, 24, 27, 33, 38, | RAD50, Hs.97530, Hs.203866, Hs.222808, |
| 4 | 10 | 4 | 0 | 2, 4, 5, 10, 11, 12, 13, 14, 18, 19, | Hs.120716, MYBL1, DCTD, ID2H, |
| 5 | 10 | 4 | 0.1 | 2, 5, 10, 12, 13, 14, 18, 19, 20, 29, | Clone.704802, MYBL1, DCTD, ID2H, |
| 6 | 10 | 4 | 0.1 | 2, 5, 10, 11, 12, 13, 14, 18, 19, 29, | MYBL1, DCTD, ID2H, PTK, |
| 7 | 10 | 4 | 0.2 | 2, 5, 10, 12, 13, 14, 18, 19, 29, 40, | Clone.704802, DCTD, ID2H, PTK, |
| 8 | 10 | 4 | 0 | 2, 5, 7, 10, 11, 12, 13, 14, 18, 19, | Hs.120716, MYBL1, ID2H, PTK, |
| 9 | 10 | 4 | 0.1 | 2, 5, 7, 10, 12, 13, 14, 18, 19, 29, | Clone.704802, MYBL1, ID2H, PTK, |
| 10 | 10 | 4 | 0.3 | 2, 7, 10, 13, 14, 18, 19, 29, 40, 46, | Clone.704802, Hs.75859, ID2H, PTK, |
| 11 | 10 | 4 | 0.4 | 9, 10, 13, 18, 19, 22, 29, 41, 45, 46, | Hs.626884, Hs.75859, ID2H, PTK, |
| 12 | 10 | 4 | 0.2 | 2, 7, 9, 13, 14, 18, 19, 22, 45, 46, | JMJD1B, Hs.75859, ID2H, PTK, |
| 13 | 10 | 4 | 0.4 | 3, 4, 11, 20, 21, 23, 24, 27, 38, 40, | Clone.1333841, Hs.203866, Hs.222808, DCTD, |
| 14 | 10 | 4 | 0.4 | 0, 8, 10, 13, 18, 22, 28, 39, 44, 46, | JNK3, TDT, Hs.75859, PTK, |
| 15 | 11 | 4 | 0 | 1, 2, 3, 7, 8, 10, 13, 14, 15, 18, 19, | Hs.75859, Hs.120716, MYBL1, PTK, |
| 16 | 10 | 4 | 0 | 0, 1, 2, 3, 8, 10, 13, 14, 18, 19, | Hs.75859, Hs.120716, DCTD, PTK, |
| 17 | 11 | 4 | 0 | 2, 3, 5, 7, 8, 10, 12, 13, 14, 18, 19, | Clone.704802, Hs.120716, MYBL1, PTK, |
| 18 | 10 | 4 | 0 | 1, 2, 7, 8, 10, 11, 13, 14, 19, 21, | KIAA1039, Hs.28355, Hs.120716, PTK, |
| 19 | 10 | 4 | 0 | 1, 2, 5, 7, 8, 10, 11, 13, 14, 19, | KIAA1039, Hs.120716, MYBL1, PTK, |
| 20 | 10 | 4 | 0 | 1, 2, 5, 8, 10, 11, 13, 14, 19, 21, | KIAA1039, Hs.120716, DCTD, PTK, |

**Table A.18. Top 20 longest patterns with Xmeans binning, $K_{max}$=20, $S_{min}$=10, ABC<50%**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 10 | 4 | 0.4 | 9, 10, 13, 18, 19, 22, 29, 41, 45, 46, | Hs.626884, Hs.75859, ID2H, PTK, |
| 2 | 10 | 4 | 0 | 1, 2, 8, 10, 11, 13, 14, 19, 21, 23, | KIAA1039, Hs.28355, DCTD, PTK, |
| 3 | 10 | 4 | 0.1 | 1, 2, 3, 8, 10, 13, 14, 18, 19, 29, | Hs.75859, MYBL1, DCTD, PTK, |
| 4 | 10 | 4 | 0.1 | 2, 5, 10, 12, 13, 14, 18, 19, 20, 29, | Clone.704802, MYBL1, DCTD, ID2H, |
| 5 | 11 | 4 | 0.09 | 2, 3, 5, 8, 10, 12, 13, 14, 18, 19, 29, | Clone.704802, MYBL1, DCTD, PTK, |
| 6 | 10 | 4 | 0.1 | 2, 5, 10, 11, 12, 13, 14, 18, 19, 29, | MYBL1, DCTD, ID2H, PTK, |
| 7 | 10 | 4 | 0.4 | 3, 4, 11, 20, 21, 23, 24, 27, 38, 40, | Clone.1333841, Hs.203866, Hs.222808, DCTD, |
| 8 | 10 | 4 | 0.2 | 2, 3, 8, 10, 13, 14, 18, 19, 29, 40, | Clone.704802, Hs.75859, DCTD, PTK, |
| 9 | 10 | 4 | 0.2 | 2, 5, 10, 12, 13, 14, 18, 19, 29, 40, | Clone.704802, DCTD, ID2H, PTK, |
| 10 | 10 | 4 | 0.4 | 2, 4, 11, 21, 22, 23, 24, 27, 33, 38, | RAD50, Hs.97530, Hs.203866, Hs.222808, |
| 11 | 10 | 4 | 0.1 | 2, 3, 7, 8, 10, 13, 14, 18, 19, 29, | Clone.704802, Hs.75859, MYBL1, PTK, |
| 12 | 10 | 4 | 0.3 | 2, 7, 10, 13, 14, 18, 19, 29, 40, 46, | Clone.704802, Hs.75859, ID2H, PTK, |
| 13 | 10 | 4 | 0.2 | 2, 7, 9, 13, 14, 18, 19, 22, 45, 46, | JMJD1B, Hs.75859, ID2H, PTK, |
| 14 | 11 | 4 | 0 | 1, 3, 7, 8, 9, 10, 13, 14, 15, 18, 19, | Hs.75859, MYBL1, MYBL1, PTK, |
| 15 | 10 | 4 | 0.1 | 2, 5, 7, 10, 12, 13, 14, 18, 19, 29, | Clone.704802, MYBL1, ID2H, PTK, |
| 16 | 11 | 3 | 0.363 | 1, 3, 9, 11, 20, 21, 22, 24, 28, 31, 38, | Clone.685761, Hs.203866, Hs.222808, |
| 17 | 11 | 3 | 0.363 | 1, 3, 9, 11, 19, 21, 22, 25, 28, 38, 39, | Clone.685761, Hs.203866, PTK, |
| 18 | 10 | 3 | 0.4 | 1, 2, 11, 19, 22, 23, 24, 25, 26, 29, | RAD50, Hs.203866, OGG1, |
| 19 | 11 | 3 | 0.363 | 1, 2, 3, 11, 19, 22, 23, 25, 26, 29, 34, | Hs.203866, OGG1, PTK, |
| 20 | 10 | 3 | 0.2 | 7, 8, 9, 10, 13, 14, 15, 18, 39, 40, | Hs.75859, OGG1, PTK, |

**Table A.19. Top 20 highest support patterns with pre-set value binning**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 28 | 1 | 0.5 | 0, 1, 3, 5, 6, 8, 9, 10, 11, 12, 14, 15, 20, 21, 24, 25, 28, 29, 31, 33, 34, 36, 37, 39, 40, 41, 42, 43 | JMJD1B, |
| 2 | 22 | 1 | 0.636 | 1, 2, 3, 8, 11, 19, 20, 22, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 43, 44 | Hs.203866, |
| 3 | 21 | 2 | 0.571 | 1, 3, 5, 10, 11, 12, 14, 20, 21, 24, 29, 31, 33, 34, 36, 37, 39, 40, 41, 42, 43 | JMJD1B, Hs.190288 |
| 4 | 20 | 1 | 0.4 | 1, 2, 3, 4, 6, 12, 13, 16, 17, 18, 19, 20, 24, 25, 27, 28, 33, 34, 45, 46 | SLCO3A1, |
| 5 | 19 | 1 | 0.526 | 1, 3, 6, 7, 14, 18, 20, 21, 23, 24, 25, 27, 31, 33, 35, 36, 42, 45, 46 | IL2RB, |
| 6 | 19 | 2 | 0.631 | 1, 2, 3, 11, 19, 20, 22, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 43 | Hs.203866, Hs.190288 |
| 7 | 19 | 1 | 0.473 | 0, 1, 2, 3, 4, 7, 16, 17, 18, 19, 24, 26, 27, 28, 33, 34, 35, 38, 45 | BCL2, |
| 8 | 18 | 2 | 0.444 | 0, 1, 3, 5, 8, 10, 11, 14, 15, 20, 25, 28, 29, 33, 36, 37, 39, 40 | Hs.4766, JMJD1B, |
| 9 | 18 | 2 | 0.555 | 0, 1, 3, 5, 11, 14, 20, 21, 24, 25, 28, 31, 33, 34, 36, 39, 41, 43 | JMJD1B, Hs.87589, |
| 10 | 18 | 2 | 0.5 | 3, 5, 6, 9, 10, 12, 14, 20, 21, 24, 25, 28, 29, 33, 36, 39, 40, 41 | JMJD1B, Hs.123318, |
| 11 | 18 | 1 | 0.166 | 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 16, 17, 22, 23, 26, 28, 34 | Hs.108327 (damage spec. DNA-binding prot. 1), |
| 12 | 18 | 1 | 0.333 | 1, 3, 4, 5, 6, 11, 13, 14, 16, 21, 22, 23, 24, 27, 34, 35, 41, 45 | Clone.1355676, |
| 13 | 17 | 2 | 0.294 | 1, 2, 3, 4, 6, 12, 13, 16, 17, 18, 19, 20, 24, 25, 27, 33, 34 | TTG2, SLCO3A1, |
| 14 | 17 | 2 | 0.588 | 0, 3, 5, 8, 9, 11, 21, 24, 25, 28, 29, 33, 36, 39, 41, 42, 43 | JMJD1B, CSNK1G2 (casein kinase g2), |
| 15 | 17 | 2 | 0.705 | 1, 8, 9, 10, 21, 24, 25, 28, 31, 34, 36, 37, 39, 40, 41, 42, 43 | JMJD1B, Clone.1333667, |
| 16 | 17 | 2 | 0.529 | 3, 6, 8, 9, 10, 11, 12, 14, 25, 29, 31, 34, 36, 39, 41, 42, 43 | JMJD1B, Clone.1372073 |
| 17 | 17 | 2 | 0.294 | 0, 1, 3, 5, 6, 8, 9, 11, 12, 14, 20, 21, 24, 25, 33, 34, 41 | JMJD1B, TTG2, |
| 18 | 17 | 2 | 0.47 | 1, 3, 6, 11, 12, 14, 15, 20, 21, 24, 31, 33, 37, 39, 40, 41, 42 | JMJD1B, BRCA2, |
| 19 | 17 | 2 | 0.352 | 1, 3, 6, 8, 9, 10, 11, 12, 14, 15, 21, 24, 25, 28, 33, 37, 40 | JMJD1B, PRKCD (prot. kinase C d), |
| 20 | 17 | 2 | 0.352 | 0, 1, 3, 5, 6, 9, 10, 11, 12, 15, 21, 25, 29, 33, 39, 41, 43 | JMJD1B, Clone.1371532, |

**Table A.20. Top 20 longest patterns with pre-set value binning (gene names ommitted)**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 2 | 257 | 0 | 16, 17 |
| 2 | 2 | 174 | 0 | 18, 19 |
| 3 | 2 | 170 | 0 | 12, 17 |
| 4 | 2 | 168 | 0 | 11, 21 |
| 5 | 2 | 165 | 0.5 | 21, 33 |
| 6 | 2 | 162 | 0 | 2, 11 |
| 7 | 2 | 161 | 0.5 | 1, 27 |
| 8 | 2 | 160 | 0 | 10, 15 |
| 9 | 2 | 158 | 0 | 12, 13 |
| 10 | 2 | 157 | 0 | 13, 17 |
| 11 | 2 | 157 | 0.5 | 11, 25 |
| 12 | 2 | 157 | 0 | 15, 16 |
| 13 | 2 | 156 | 0 | 5, 10 |
| 14 | 2 | 156 | 0 | 7, 11 |
| 15 | 2 | 156 | 1 | 36, 39 |
| 16 | 2 | 154 | 1 | 24, 27 |
| 17 | 2 | 154 | 0 | 11, 19 |
| 18 | 2 | 154 | 1 | 30, 31 |
| 19 | 2 | 152 | 0 | 12, 16 |
| 20 | 2 | 152 | 0 | 7, 18 |

**Table A.21. Top 20 highest support patterns with pre-set value binning, with 100% ABC or GCB support**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 13 | 2 | 0 | 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 16, 17 | Hs.120716, Hs.108327 (damage spec. DNA-binding prot. 1), |
| 2 | 13 | 1 | 1 | 24, 27, 28, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39 | JAW1, |
| 3 | 12 | 3 | 0 | 1, 2, 3, 4, 6, 12, 13, 16, 17, 18, 19, 20 | Hs.120716, TTG2, SLCO3A1, |
| 4 | 12 | 3 | 0 | 0, 1, 3, 5, 6, 8, 9, 11, 12, 14, 20, 21 | JMJD1B, Hs.120716, TTG2, |
| 5 | 12 | 4 | 0 | 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 16, 17 | Hs.120716, TTG2, TTG2, Hs.108327 (damage spec. DNA-binding prot. 1), |
| 6 | 12 | 2 | 0 | 0, 4, 7, 8, 9, 10, 11, 15, 16, 18, 20, 21 | Hs.120716, LCK, |
| 7 | 12 | 3 | 0 | 0, 5, 6, 8, 9, 10, 12, 13, 14, 15, 16, 17 | PTP-IVA, Hs.120716, Clone.713158, |
| 8 | 12 | 2 | 0 | 0, 1, 2, 7, 9, 11, 12, 14, 15, 16, 17, 19 | Hs.120716, BCL2, |
| 9 | 11 | 4 | 0 | 2, 3, 4, 6, 12, 13, 16, 17, 18, 19, 20 | Hs.120716, TTG2, TTG2, SLCO3A1, |
| 10 | 11 | 4 | 0 | 0, 1, 3, 5, 6, 9, 10, 11, 14, 15, 20 | JMJD1B, Hs.120716, Hs.224323, FMR2, |
| 11 | 11 | 4 | 0 | 0, 3, 5, 6, 8, 9, 11, 12, 14, 20, 21 | JMJD1B, Hs.120716, TTG2, TTG2, |
| 12 | 11 | 4 | 0 | 3, 5, 6, 8, 9, 10, 11, 12, 14, 15, 21 | JMJD1B, Hs.120716, Hs.202588, Hs.106771, |
| 13 | 11 | 3 | 0 | 0, 1, 3, 5, 6, 9, 10, 11, 12, 15, 21 | JMJD1B, Hs.120716, Clone.1371532, |
| 14 | 11 | 2 | 0 | 2, 3, 5, 6, 9, 11, 12, 15, 16, 22, 23 | Clone.1371532, Hs.108327 (damage spec. DNA-binding prot. 1), |
| 15 | 11 | 4 | 0 | 2, 3, 4, 5, 6, 9, 11, 14, 17, 22, 23 | TTG2, TTG2, SA3np, Hs.108327 (damage spec. DNA-binding prot. 1), |
| 16 | 11 | 3 | 0 | 2, 3, 4, 5, 9, 11, 12, 14, 15, 16, 17 | Hs.120716, ALU subfamily SB, Hs.108327 (damage spec. DNA-binding prot. 1), |
| 17 | 11 | 2 | 0 | 2, 5, 6, 8, 9, 12, 14, 15, 16, 17, 23 | PTP-IVA, Hs.108327 (damage spec. DNA-binding prot. 1), |
| 18 | 11 | 2 | 0 | 2, 4, 5, 8, 9, 11, 12, 14, 15, 22, 23 | Hs.108327 (damage spec. DNA-binding prot. 1), MZF1, |
| 19 | 11 | 2 | 1 | 24, 27, 28, 30, 31, 32, 33, 34, 35, 38, 39 | JAW1, Hs.87589, |
| 20 | 11 | 2 | 1 | 24, 27, 29, 30, 31, 32, 33, 34, 37, 38, 39 | JAW1, Hs.190288 |

**Table A.22. Top 20 highest support patterns with pre-set value binning, and $G_{min}$=8**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 8 | 8 | 0 | 3, 4, 13, 16, 17, 18, 19, 20 | Hs.120716, ALU subfamily SB, Hs.136345, Hs.224323, Clone.825199, TTG2, TTG2, SLCO3A1, |
| 2 | 8 | 8 | 0.375 | 2, 3, 4, 17, 19, 24, 27, 34 | Clone.684852, TTG2, TTG2, SLCO3A1, Clone.2017, Clone.2022, BCL2, ALU class C, |
| 3 | 8 | 8 | 0.375 | 2, 3, 4, 18, 19, 24, 27, 34 | Clone.684852, TTG2, TTG2, SLCO3A1, Clone.2017, AIbp63, BCL2, Hs.190288 |
| 4 | 8 | 8 | 0.375 | 2, 3, 4, 16, 19, 24, 27, 34 | TTG2, TTG2, SLCO3A1, Clone.2017, APR, Clone.2022, AIbp63, BCL2, |
| 5 | 8 | 8 | 0 | 1, 2, 12, 13, 16, 17, 18, 19 | CD21, PTP-IVA, Hs.120716, TTG2, SLCO3A1, Hs.120751, BCL2, MAPKKK5, |
| 6 | 8 | 8 | 0.25 | 8, 9, 11, 12, 14, 21, 25, 33 | JMJD1B, TTG2, TTG2, Hs.106771, Clone.1357367, POU4F1, PRKCD (prot. kinase C d), PCKD, |
| 7 | 8 | 8 | 0.5 | 1, 2, 11, 19, 33, 38, 39, 41 | RAD50, Hs.203866, MEK1, Clone.1371532, Hs.87589, Hs.120751, MAPKKK5, Hs.190288 |
| 8 | 8 | 8 | 0.5 | 2, 11, 19, 22, 30, 33, 39, 41 | RAD50, Hs.203866, Clone.1371532, Hs.87589, Hs.46913, MAPKKK5, Hs.190288 Hs.24724, |
| 9 | 8 | 8 | 0.5 | 2, 11, 19, 22, 30, 33, 38, 41 | RAD50, Hs.203866, Clone.1371532, Hs.87589, IL10RB, MAPKKK5, Hs.190288 Hs.24724, |
| 10 | 8 | 8 | 0.5 | 2, 11, 19, 22, 30, 38, 39, 41 | RAD50, Hs.203866, Clone.1371532, Hs.87589, MAPKKK5, HKRT1, Hs.190288 Hs.24724, |
| 11 | 8 | 8 | 0.375 | 2, 11, 18, 19, 21, 24, 33, 41 | RAD50, TTG2, TTG2, SRPK2, Hs.46913, Hs.120751, CD44, Hs.190288 |
| 12 | 8 | 8 | 0.25 | 2, 11, 18, 19, 21, 23, 33, 41 | RAD50, TTG2, TTG2, Clone.1371532, SRPK2, IL10RB, CD44, Hs.190288 |
| 13 | 8 | 8 | 0.25 | 2, 11, 18, 19, 21, 23, 24, 41 | RAD50, WASPIP, Hs.192738, TTG2, TTG2, SRPK2, CD44, Hs.190288 |
| 14 | 8 | 8 | 0.25 | 2, 4, 11, 19, 21, 23, 24, 41 | RAD50, Hs.193796, Hs.192738, TTG2, TTG2, SRPK2, CD44, Hs.190288 |
| 15 | 8 | 8 | 0.25 | 2, 4, 11, 18, 21, 23, 24, 41 | RAD50, Hs.192738, TTG2, TTG2, SRPK2, CD44, Hs.190288 TNK1, |
| 16 | 8 | 8 | 0.125 | 0, 9, 11, 16, 18, 21, 22, 33 | Clone.1270568, ALU subfamily SX, TTG2, TTG2, KIAA0151, LCK, Clone.1371532, Hs.46913, |
| 17 | 8 | 8 | 0.125 | 0, 7, 9, 11, 18, 21, 22, 33 | Clone.1270568, ALU subfamily SX, Clone.685761, TTG2, TTG2, LCK, Clone.1371532, Hs.46913, |
| 18 | 8 | 8 | 0.125 | 7, 9, 11, 16, 18, 21, 22, 33 | Clone.1270568, ALU subfamily SX, TTG2, TTG2, LCK, Hs.226955, Clone.1371532, Hs.46913, |
| 19 | 8 | 8 | 0.125 | 0, 7, 11, 16, 18, 21, 22, 33 | Clone.1270568, ALU subfamily SX, TTG2, TTG2, LCK, Clone.1371532, Hs.46913, Clone.1288046, |
| 20 | 8 | 8 | 0.375 | 0, 3, 11, 19, 20, 25, 27, 33 | Clone.685761, BCL6, BCL6, TTG2, TTG2, Hs.87589, Hs.192047, Hs.123294, |

**Table A.23. Top 20 highest support patterns with pre-set value binning, and $G_{min}$=12**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 7 | 12 | 0.142 | 2, 11, 18, 19, 21, 23, 41 | RAD50, WASPIP, WASPIP, WASPIP, Hs.192738, TTG2, TTG2, Clone.1371532, SRPK2, IL10RB, CD44, Hs.190288 |
| 2 | 7 | 12 | 0 | 0, 3, 7, 16, 17, 18, 19 | CSNK1G2, Hs.120716, ALU subfamily SB, Clone.825199, TTG2, TTG2, Clone.2017, Hs.192047, BCL2, Hs.59368, Hs.123294, Clone.1288046, |
| 3 | 7 | 12 | 0 | 5, 9, 10, 14, 15, 16, 17 | Clone.814651, KIAA1037, PTP-IVA, Hs.120716, Hs.224323, Clone.825199, ALU subfamily SB, CD10, PTP1B, PTP1B, Clone.713158, KIAA0805, |
| 4 | 7 | 13 | 0 | 5, 10, 13, 14, 15, 16, 17 | Clone.814651, KIAA1037, Hs.97275, PTP-IVA, Hs.120716, Hs.224323, Clone.825199, ALU subfamily SB, Hs.88102, CD10, Clone.713158, KIAA0805, KCNA3, |
| 5 | 7 | 12 | 0 | 5, 9, 10, 13, 14, 16, 17 | Clone.814651, KIAA1037, PTP-IVA, Clone.1351325, Hs.120716, Hs.224323, Clone.825199, ALU subfamily SB, CD10, Clone.713158, KIAA0805, ALU subfamily SQ, |
| 6 | 6 | 12 | 0.5 | 1, 7, 18, 27, 33, 45 | RAD50, Clone.685761, Clone.826377, IL4R, Clone.1371532, Hs.192047, Hs.120751, BCL2, MAPKKK5, STK11, IL2RB, IL2RB, |
| 7 | 6 | 13 | 0.666 | 1, 7, 27, 31, 33, 45 | Hs.27774, Hs.125815, Clone.685761, MGC5178, BRCA2, Hs.58297, Hs.87589, Hs.192047, Hs.120751, MAPKKK5, STK11, IL2RB, IL2RB, |
| 8 | 6 | 12 | 0.333 | 1, 3, 7, 21, 27, 33 | KIAA1037, Clone.685761, TTG2, ALU subfamily J, BRCA2, IL4R, Clone.1371532, Hs.87589, Hs.123254, ID2H, IL2RB, IL2RB, |
| 9 | 6 | 12 | 0.666 | 7, 21, 27, 31, 45, 46 | Hs.125815, Hs.228201, BRCA2, Hs.87589, Hs.46913, SC5DL, SLA, Clone.1369049, INPPL1, IL2RB, IL2RB, |
| 10 | 6 | 12 | 0.5 | 1, 7, 21, 27, 31, 33 | Clone.1340461, Hs.125815, Clone.685761, Clone.1357367, MGC5178, BRCA2, Hs.87589, Hs.120751, Hs.123254, SP100, IL2RB, IL2RB, |
| 11 | 6 | 12 | 0.333 | 6, 18, 20, 21, 27, 33 | KIAA1037, Hs.136204, Clone.685761, TTG2, TTG2, SRPK2, PTP1B, PTP1B, IL2RB, IL2RB, Hs.136589, Hs.128564, |
| 12 | 6 | 13 | 0.5 | 1, 7, 21, 27, 33, 45 | RAD50, Hs.125815, Clone.685761, MGC5178, BRCA2, Clone.826377, IL4R, Clone.1371532, Hs.87589, Hs.120751, DCTD, IL2RB, IL2RB, |
| 13 | 6 | 12 | 0.5 | 1, 7, 23, 27, 33, 45 | RAD50, Hs.125815, MGC5178, Clone.826377, IL4R, Hs.58297, Clone.1371532, Hs.87589, Hs.192047, DCTD, IL2RB, IL2RB, |
| 14 | 6 | 14 | 0.333 | 1, 7, 21, 23, 27, 33 | RAD50, KIAA1037, Hs.125815, TTG2, Clone.1357367, MGC5178, ALU subfamily J, Clone.826377, IL4R, Clone.1371532, Hs.87589, DCTD, IL2RB, IL2RB, |
| 15 | 6 | 12 | 0.166 | 1, 7, 18, 21, 23, 33 | RAD50, KIAA1037, Clone.1340456, TTG2, Clone.1355520, ALU subfamily J, Hs.226955, Clone.826377, IL4R, Clone.1371532, IL2RB, IL2RB, |
| 16 | 6 | 12 | 0.666 | 1, 7, 25, 27, 33, 45 | Hs.27774, RAD50, Hs.125815, Clone.685761, Clone.1371532, Hs.87589, Hs.192047, Hs.120751, MAPKKK5, DCTD, STK11, IL2RB, |
| 17 | 6 | 13 | 0.666 | 1, 7, 25, 27, 31, 45 | Hs.27774, Hs.125815, HIP1R, Clone.685761, Hs.173108, Hs.87589, SC5DL, Hs.192047, Hs.120751, MAPKKK5, STK11, IL2RB, ALU class C, |
| 18 | 6 | 13 | 0.5 | 1, 7, 21, 24, 25, 33 | RAD50, Hs.125815, Clone.685761, TTG2, Clone.1355520, CD27, Hs.226955, Hs.87589, Hs.120751, SP100, Clone.1288046, ID2H, IL2RB, |
| 19 | 6 | 12 | 0.666 | 7, 21, 24, 25, 27, 33 | RAD50, Hs.125815, Clone.685761, TTG2, TTG2, Hs.87589, Hs.120751, TNRF2, PTP1B, SP100, ID2H, IL2RB, |
| 20 | 6 | 13 | 0.5 | 1, 7, 21, 25, 27, 33 | RAD50, KIAA1037, Hs.125815, Clone.685761, TTG2, Clone.1357367, Clone.1371532, Hs.87589, Hs.120751, DCTD, SP100, ID2H, IL2RB, |

**Table A.24. Top 20 longest patterns with pre-set value binning (gene names ommitted), $S_{min}$=5**

| | | | | |
|---|---|---|---|---|
| 1 | 5 | 32 | 0 | 10, 14, 15, 16, 17 |
| 2 | 5 | 32 | 0 | 10, 13, 15, 16, 17 |
| 3 | 5 | 31 | 0 | 5, 10, 15, 16, 17 |
| 4 | 5 | 31 | 0 | 12, 13, 14, 16, 17 |
| 5 | 5 | 30 | 0 | 12, 14, 15, 16, 17 |
| 6 | 5 | 30 | 0.2 | 2, 11, 18, 21, 41 |
| 7 | 5 | 30 | 0 | 5, 10, 14, 16, 17 |
| 8 | 5 | 30 | 0 | 12, 13, 15, 16, 17 |
| 9 | 5 | 29 | 0 | 5, 14, 15, 16, 17 |
| 10 | 5 | 29 | 0.2 | 2, 11, 18, 19, 41 |
| 11 | 5 | 29 | 0 | 7, 10, 15, 16, 18 |
| 12 | 5 | 28 | 0 | 12, 13, 16, 17, 18 |
| 13 | 5 | 28 | 0 | 7, 13, 16, 17, 18 |
| 14 | 5 | 28 | 0.6 | 11, 21, 24, 25, 33 |
| 15 | 5 | 28 | 0 | 2, 11, 18, 19, 21 |
| 16 | 5 | 28 | 0 | 10, 12, 15, 16, 17 |
| 17 | 5 | 28 | 0 | 6, 10, 15, 16, 17 |
| 18 | 5 | 28 | 0 | 5, 10, 14, 15, 16 |
| 19 | 5 | 28 | 0 | 10, 13, 14, 16, 17 |
| 20 | 5 | 28 | 0 | 13, 14, 15, 16, 17 |

**Table A.25. Top 20 longest patterns with pre-set value binning (gene names ommitted), $S_{min}$=5, ABC>50%**

| Rank | Support | #Genes | %ABC | Supporting Tissues |
|---|---|---|---|---|
| 1 | 5 | 28 | 0.6 | 11, 21, 24, 25, 33 |
| 2 | 5 | 25 | 0.6 | 11, 21, 31, 33, 39 |
| 3 | 5 | 24 | 0.6 | 11, 21, 25, 28, 33 |
| 4 | 5 | 23 | 0.6 | 6, 21, 24, 25, 33 |
| 5 | 5 | 23 | 0.6 | 11, 21, 24, 25, 29 |
| 6 | 5 | 23 | 0.6 | 11, 21, 25, 31, 33 |
| 7 | 5 | 23 | 0.6 | 11, 21, 24, 33, 46 |
| 8 | 5 | 23 | 0.6 | 11, 19, 25, 27, 33 |
| 9 | 5 | 22 | 0.8 | 21, 24, 25, 33, 46 |
| 10 | 5 | 22 | 1 | 31, 36, 37, 39, 40 |
| 11 | 5 | 22 | 0.6 | 11, 21, 24, 29, 33 |
| 12 | 5 | 22 | 0.6 | 11, 21, 25, 29, 33 |
| 13 | 5 | 22 | 0.6 | 11, 21, 31, 33, 46 |
| 14 | 5 | 21 | 0.6 | 1, 21, 24, 25, 33 |
| 15 | 5 | 21 | 0.6 | 1, 21, 24, 25, 27 |
| 16 | 5 | 21 | 0.6 | 20, 21, 24, 25, 27 |
| 17 | 5 | 21 | 0.6 | 2, 19, 24, 27, 34 |
| 18 | 5 | 21 | 1 | 36, 37, 39, 40, 43 |
| 19 | 5 | 21 | 1 | 29, 36, 39, 42, 43 |
| 20 | 5 | 21 | 0.8 | 21, 24, 25, 28, 33 |

**Table A.26. Top 20 longest patterns with pre-set value binning, $S_{min}$=7**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 7 | 13 | 0 | 5, 10, 13, 14, 15, 16, 17 | Clone.814651, KIAA1037, Hs.97275, PTP-IVA, Hs.120716, Hs.224323, Clone.825199, ALU subfamily SB, Hs.88102, CD10, Clone.713158, KIAA0805, KCNA3, |
| 2 | 7 | 12 | 0.142 | 2, 11, 18, 19, 21, 23, 41 | RAD50, WASPIP, WASPIP, WASPIP, Hs.192738, TTG2, TTG2, Clone.1371532, SRPK2, IL10RB, CD44, Hs.190288 |
| 3 | 7 | 12 | 0 | 0, 3, 7, 16, 17, 18, 19 | CSNK1G2, Hs.120716, ALU subfamily SB, Clone.825199, TTG2, TTG2, Clone.2017, Hs.192047, BCL2, Hs.59368, Hs.123294, Clone.1288046, |
| 4 | 7 | 12 | 0 | 5, 9, 10, 14, 15, 16, 17 | Clone.814651, KIAA1037, PTP-IVA, Hs.120716, Hs.224323, Clone.825199, ALU subfamily SB, CD10, PTP1B, PTP1B, Clone.713158, KIAA0805, |
| 5 | 7 | 12 | 0 | 5, 9, 10, 13, 14, 16, 17 | Clone.814651, KIAA1037, PTP-IVA, Clone.1351325, Hs.120716, Hs.224323, Clone.825199, ALU subfamily SB, CD10, Clone.713158, KIAA0805, ALU subfamily SQ, |
| 6 | 7 | 11 | 0.428 | 2, 3, 4, 19, 24, 27, 34 | Clone.684852, TTG2, TTG2, SLCO3A1, Clone.2017, APR, Clone.2022, Albp63, BCL2, Hs.190288 ALU class C, |
| 7 | 7 | 11 | 0.142 | 8, 9, 11, 12, 14, 21, 25 | JMJD1B, Hs.130721, TTG2, TTG2, Hs.202588, Hs.106771, Clone.1357367, POU4F1, PRKCD (prot. kinase C d), PCKD, ALU class C, |
| 8 | 7 | 11 | 0.428 | 2, 11, 19, 22, 38, 39, 41 | RAD50, WASPIP, WASPIP, Hs.203866, Clone.1371532, Clone.1336373, Hs.87589, MAPKKK5, HKRT1, Hs.190288 Hs.24724, |
| 9 | 7 | 11 | 0.285 | 2, 11, 18, 19, 21, 33, 41 | RAD50, TTG2, TTG2, Clone.1371532, SRPK2, Hs.46913, IL10RB, Hs.120751, GENE88X__STAT3_APRF_acute_phase_response_factor__Clone_1358208, CD44, Hs.190288 |
| 10 | 7 | 11 | 0.285 | 2, 11, 18, 19, 21, 24, 33 | RAD50, Clone.685761, TTG2, TTG2, Hs.226955, SRPK2, Hs.46913, Hs.120751, PTP1B, CD44, Hs.190288 |
| 11 | 7 | 11 | 0.285 | 2, 11, 19, 21, 23, 24, 41 | RAD50, Hs.193796, WASPIP, Hs.192738, TTG2, TTG2, SRPK2, Hs.87589, HKRT1, CD44, Hs.190288 |
| 12 | 7 | 11 | 0.142 | 1, 2, 11, 18, 19, 21, 41 | RAD50, CD21, CD21, Hs.61506, BDP1, TTG2, Clone.1371532, Clone.1336373, Hs.120751, CD44, Hs.190288 |
| 13 | 7 | 11 | 0 | 3, 10, 11, 15, 18, 19, 20 | Clone.1339726, BCL6, Hs.120716, ALU subfamily SB, Hs.224323, Clone.825199, FMR2, Clone.1336373, ERK3, Hs.123294, Hs.24724, |
| 14 | 7 | 11 | 0.428 | 0, 3, 11, 19, 25, 27, 33 | ARHGEF1, Clone.685761, BCL6, BCL6, Hs.124049, TTG2, TTG2, Clone.1371532, Hs.87589, Hs.192047, Hs.123294, |
| 15 | 7 | 11 | 0 | 0, 3, 11, 14, 18, 19, 20 | CSNK1G2, BCL6, Hs.120716, ALU subfamily SB, Hs.224323, Clone.825199, FMR2, BDP1, TTG2, TTG2, GENE353X__Unknown__UG_Hs_164617__ESTs__Clone_1351973_, |
| 16 | 7 | 11 | 0.428 | 2, 7, 18, 19, 24, 27, 33 | RAD50, Clone.685761, TTG2, TTG2, SRPK2, Hs.46913, Hs.120751, BCL2, PTP1B, Hs.123294, Hs.24724, |
| 17 | 7 | 11 | 0 | 3, 4, 7, 16, 17, 18, 19 | Hs.120716, ALU subfamily SB, Hs.136345, Clone.825199, TTG2, TTG2, Clone.2017, ERK3, BCL2, Hs.59368, Clone.1288046, |
| 18 | 7 | 11 | 0 | 5, 10, 12, 14, 15, 16, 17 | GENE144X__Unknown__Clone_1356654_, Clone.685761, PTP-IVA, Hs.120716, ALU subfamily SB, Hs.88102, CD10, PTP1B, PTP1B, Clone.713158, KIAA0805, |
| 19 | 7 | 11 | 0 | 9, 10, 12, 14, 15, 16, 17 | Clone.1270568, Clone.826541, PTP-IVA, Hs.120716, ALU subfamily SB, CD10, PTP1B, PTP1B, Clone.713158, KIAA0805, SP100, |
| 20 | 7 | 11 | 0 | 9, 10, 13, 14, 15, 16, 17 | Clone.814651, KIAA1037, PTP-IVA, Hs.120716, ALU subfamily SB, Hs.224323, Clone.825199, ALU subfamily SB, CD10, Clone.713158, KIAA0805, |

**Table A.27. Top 20 longest patterns with pre-set value binning, $S_{min}$=7, ABC>50%**

| Rank | Support | #Genes | %ABC | Supporting Tissues | Genes |
|---|---|---|---|---|---|
| 1 | 7 | 10 | 0.714 | 7, 21, 24, 27, 31, 45, 46 | Hs.125815, Hs.228201, BRCA2, Hs.87589, Hs.46913, SC5DL, SLA, INPPL1, IL2RB, |
| 2 | 7 | 10 | 0.571 | 1, 7, 21, 24, 27, 31, 45 | Hs.125815, Clone.685761, MGC5178, BRCA2, Hs.87589, SC5DL, INPPL1, IL2RB, ALU class C, |
| 3 | 7 | 10 | 0.714 | 7, 21, 24, 27, 31, 33, 45 | Hs.125815, Clone.685761, MGC5178, BRCA2, SRPK2, Hs.87589, Hs.46913, Hs.120751, SLA, IL2RB, |
| 4 | 7 | 10 | 0.571 | 1, 11, 21, 24, 25, 29, 33 | RAD50, JMJD1B, KIAA0151, CD27, Hs.226955, Hs.120751, SP100, Clone.1288046, ID2H, Hs.128564, |
| 5 | 7 | 10 | 0.714 | 11, 21, 24, 25, 28, 29, 33 | JMJD1B, CSNK1G2 (casein kinase g2), KIAA0151, PCKD, PCKD, Clone.1671615, HNPP, SP100, Clone.1288046, ID2H, |
| 6 | 7 | 10 | 0.571 | 1, 11, 21, 24, 25, 31, 33 | JMJD1B, Hs.125815, Clone.685761, KIAA0151, CD27, Hs.226955, Hs.87589, Hs.120751, SP100, Hs.128564, |
| 7 | 7 | 10 | 0.571 | 1, 11, 21, 24, 31, 33, 39 | JMJD1B, Clone.685761, KIAA0151, MGC5178, BRCA2, Hs.226955, Hs.87589, Hs.120751, Hs.190288 Hs.128564, |
| 8 | 7 | 10 | 0.571 | 1, 11, 21, 24, 29, 31, 33 | JMJD1B, KIAA0151, CD27, MGC5178, Hs.226955, Hs.120751, Hs.190288 SP100, Hs.136589, Hs.128564, |
| 9 | 7 | 10 | 0.571 | 1, 11, 21, 24, 29, 31, 39 | JMJD1B, KIAA0151, MGC5178, Hs.226955, SC5DL, Hs.190288 TNK1, Hs.128564, ALU class C, |
| 10 | 7 | 10 | 0.571 | 1, 11, 21, 25, 29, 31, 39 | JMJD1B, KIAA0151, MEK1, Hs.226955, Clone.1336373, SC5DL, Clone.417048, Hs.128564, ALU class C, |
| 11 | 7 | 9 | 0.714 | 1, 7, 25, 27, 31, 33, 45 | Hs.27774, Hs.125815, Clone.685761, Hs.87589, Hs.192047, Hs.120751, MAPKKK5, STK11, IL2RB, |
| 12 | 7 | 9 | 0.571 | 1, 7, 21, 24, 25, 27, 33 | RAD50, Hs.125815, Clone.685761, TTG2, Hs.87589, Hs.120751, SP100, ID2H, IL2RB, |
| 13 | 7 | 9 | 0.571 | 1, 20, 21, 24, 27, 31, 46 | Clone.825338, BRCA2, Hs.87589, SC5DL, Hs.190288 Clone.1372597, INPPL1, IL2RB, Hs.136589, |
| 14 | 7 | 9 | 0.571 | 1, 20, 21, 24, 25, 27, 31 | Hs.136204, Clone.825338, Clone.685761, ZFM1, Hs.87589, SC5DL, Clone.1372597, IL2RB, Hs.128564, |
| 15 | 7 | 9 | 0.571 | 7, 21, 23, 25, 31, 36, 46 | Clone.1338156, Hs.181384, Clone.684877, Clone.1357367, POU4F1, Hs.87589, HKRT1, Clone.417048, IL2RB, |
| 16 | 7 | 9 | 0.857 | 21, 24, 27, 31, 33, 45, 46 | Hs.125815, BRCA2, Hs.87589, Hs.46913, Hs.120751, Clone.1671615, SLA, IL2RB, Hs.136589, |
| 17 | 7 | 9 | 0.571 | 6, 18, 20, 24, 25, 27, 33 | Hs.136204, Clone.685761, TTG2, TTG2, SLCO3A1, PTP1B, FLICE, IL2RB, Hs.128564, |
| 18 | 7 | 9 | 0.714 | 1, 21, 24, 27, 31, 33, 45 | Hs.125815, Clone.685761, MGC5178, BRCA2, Hs.87589, Hs.120751, IL2RB, Hs.136589, Hs.128564, |
| 19 | 7 | 9 | 0.571 | 2, 18, 19, 24, 27, 33, 34 | RAD50, TTG2, TTG2, SLCO3A1, Hs.46913, BCL2, PTP1B, Hs.190288 Hs.123294, |
| 20 | 7 | 9 | 0.571 | 2, 18, 19, 25, 27, 33, 34 | RAD50, TTG2, TTG2, SLCO3A1, MAPKKK5, PTP1B, PTP1B, GENE88X__STAT3_APRF_acute_phase_response_factor_ _Clone_1358208, Hs.123294, |