THE SIMCAL FAMILY OF ALGORITHMS FOR ANALYSIS OF
MICROARRAY DATA

by

Daniel Dvorkin

B.S., Mathematics, Metropolitan State College of Denver, 2001

A thesis submitted to the
University of Colorado at Denver and Health Sciences Center
in partial fulfillment
of the requirements for the degree of
Master of Science
Computer Science and Engineering
2005
This thesis for the Master of Science degree by
Daniel Dvorkin
has been approved by
Krzysztof Cios
Il Kyeun Ra
Harvey Greenberg

Date
Dvorkin, Daniel (M.S., Computer Science and Engineering)
The SiMCAL Family of Algorithms for Analysis of Microarray Data
Thesis directed by Professor Krzysztof Cios

ABSTRACT

SiMCAL 1, 2, and 3 (Simple Multilevel Clustering And Linking, versions 1, 2, and 3) are novel clustering algorithms for the analysis of microarray data. The purpose of these algorithms is to present complete feature sets not found in either Jarvis-Patrick clustering, from which the original SiMCAL concept is derived, or in other popular clustering methods such as hierarchical and k-means. Although each algorithm in the SiMCAL family has distinct features and methods, they all share the following attributes: they are simple, in that they are computationally inexpensive; they are multilevel, in that they provide a small number of clearly defined hierarchical levels of clusters; and they offer linking between clusters at the same level in each hierarchy. Presented here are the design, development, and analysis of the algorithms; their applications to two types of microarray data, one involving the phosphatidylinerseine receptor (PSR) and the other involving cystic fibrosis (CF); a description of the Web-based interface for visualization of results; and possible avenues for further development.

Code and data are available at http://www.dvorkin.com/daniel/Simcal123.zip under an open-source license.
This abstract accurately represents the content of the candidate’s thesis. I recommend its publication.

Signed

Krzysztof Cios
DEDICATION

For my friends and family, who have been there with me through the worst and the best; this is at least as much your accomplishment as it is mine.
ACKNOWLEDGMENT

Dr. Valerie Fadok of National Jewish Medical and Research Center was the originator of the project and supplied the data and biological background, and her belief in the merits of the project has spurred its continuing development. Dr. Krzysztof Cios of the University of Colorado at Denver and Health Sciences Center has overseen the project since its inception. Drs. Peter Henson and Jenifer Monks, both of NJMRC, have given generously of their time and expertise at critical points in development and analysis. Thami Rachidi of UCD/HSC provided major assistance with data interpretation. All of these individuals have the author’s profound thanks and gratitude.
CONTENTS

Figures ................................................. x
Tables ................................................. xi

Chapter
1. Introduction ........................................... 1
  1.1 Computational Background ............................. 1
  1.1.1 Existing Clustering Methods ....................... 1
  1.1.2 SiMCAL vs. Existing Methods ....................... 4
  1.1.3 Project History .................................. 7
1.2 Biological Background ................................. 8
  1.2.1 The Phosphatidylserine Receptor (PSR) ............. 9
  1.2.1.1 Significance of PS and PSR ..................... 9
  1.2.1.2 The PS/PSR Mechanism .......................... 10
  1.2.1.3 Avenues of PS/PSR Research ..................... 12
  1.2.2 Cystic Fibrosis (CF) .............................. 14
1.3 Notations and Definitions .............................. 15
2. SiMCAL 1 .......................................... 19
  2.1 Data and Preprocessing ............................... 19
  2.1.1 Raw Measurements and Reliability .................. 20
  2.1.2 Combination and Normalization ..................... 21
  2.1.3 Gene Expression Calculation ....................... 23
  2.2 Clustering and Linking .............................. 24
FIGURES

Figure

1.1 Simple dendrogram showing cuts at each level. ..................... 3
2.1 Expression levels of \(x\), \(y\), and \(z\). .......................... 26
2.2 Expression levels of \(x\) and \(y\). ................................. 28
2.3 Expression plot for \(G_{4355}\) and its clusters. .................... 35
2.4 Part of the introductory screen. ................................. 37
2.5 Navigation through cluster relationships. .......................... 38
2.6 \(C_{2,84}\) and its constituent genes. ............................... 40
3.1 SiMCAL 2 search interface. .................................... 59
3.2 All genes in \(\text{DchipPM\_MM}\). .................................. 60
3.3 Clusters and genes in \(\text{DchipPM\_MM}\). .......................... 61
3.4 Execution times for SiMCAL 2 components. ....................... 62
3.5 Execution times for SiMCAL 2 overall. ............................ 64
4.1 A gene from simulated SiMCAL 3 data, \(m = 8\), with its clusters. . 75
TABLES

Table

2.1 $1 - \rho$ vs. $\Delta_2$. ......................................................... 27  
3.1 SiMCAL 2 accuracy figures. .................................................. 58  
3.2 SiMCAL 2 run times, in seconds. ......................................... 63  
3.3 Total genes under consideration, and differential genes discovered. 65  
3.4 Genes shared by datasets after preprocessing. .......................... 66  
3.5 Differentially expressed genes shared by datasets. .................. 66
1. **Introduction**

This chapter covers the computational background, including an overview of existing clustering methods as opposed to SiMCAL, as well as a project history; and the biological background, with explanations of the significance of analyzing both PSR and CF data. Also, a number of notations and definitions which will be useful in later chapters are introduced.

1.1 **Computational Background**

Clustering has a rich history in the analysis of microarray data. It is an undirected method of data mining which attempts to find “clusters,” or groups of data elements, which represent some underlying structure in the data by means of an affinity between the data elements in any one cluster[4, 375-430]. A variety of clustering methods exist, all of which represent various tradeoffs between precision, reproducibility, and temporal and spatial computational requirements. In many cases, the choice of methods is fairly arbitrary.

This section looks at two of the most common existing clustering algorithms, and examines their advantages and disadvantages. A comparison with the SiMCAL algorithm family is then offered, with an explanation of the advantages of this approach. Finally, the evolution of the SiMCAL project is described.

1.1.1 **Existing Clustering Methods**

One of the most popular methods of clustering is k-means[4, 382-384]. This method requires the user to choose an arbitrary number, $k$, of *cluster centers*. Cluster centers are not data elements, but they closely resemble them, in that
they have values which place them within the data space. Usually, cluster center locations are randomly chosen at the beginning of the algorithm. The algorithm then uses an iterative process to fit the cluster centers as closely as possible to the data.

K-means has the advantage of being computationally inexpensive — it is $O(i kn)$, where $i$ is the number of iterations (a maximum value also being specified by the user in advance), $k$ is the number of clusters, and $n$ is the number of data elements. Because generally, $i, k \ll n$, this approaches $O(n)$ performance. Its disadvantages are: first, it requires the user to specify the number of clusters in advance, regardless of whether that number bears any relation to the actual structure of the data; and second, it produces convex (elliptical or cubical) clusters which, again, may not fit the data well.

Hierarchical clustering [4, 379-382] is another popular method. It is an agglomerative technique in which initially, each data element is assigned to its own cluster. Then links are built between the points to form new clusters, which are linked with existing clusters (whether single-point clusters representing the original data, or multi-point clusters built in previous steps) in a stepwise fashion. Generally, at each step, the clusters which are linked are the two which are closest to each other. The end result is a dendrogram, similar to a family tree, showing the relationship between points and the clusters they form.

The main advantage of hierarchical clustering is that the dendrogram can be “cut” at any level of resolution to determine the number and characteristics of the clusters at that level. In Figure 1.1, by cut number, the clusters are:

1. Two clusters: \{x1, x2\} and \{x3, x4, x5, x6\}. 

2. Three clusters: \{x_1, x_2\}, \{x_3\}, and \{x_4, x_5, x_6\}.

3. Four clusters: \{x_1, x_2\}, \{x_3\}, \{x_4\}, and \{x_5, x_6\}.

4. Six clusters: each data element has its own cluster.

Clearly, this offers an advantage over k-means, in that the number of clusters is not fixed ahead of time, but grows naturally out of the data. However, the number of clusters can also offer a disadvantage. For large data sets, many cut levels are available, and choosing between them may be impossible. Interpretation of Figure 1.1 is simple enough; but how best to interpret a dendrogram with thousands of data elements at the base, and hundreds or thousands of possible cut levels?

A related and perhaps more serious problem is that of computational expense. The computational complexity of hierarchical methods is generally given as \(O(n^2)\), which is already significant for large data sets. However, this is an optimistic figure, because each step of the algorithm involves considerable amounts of calculation. In other words, \(n^2\) is a lower bound on complexity; thus hierarchical
clustering is actually $\Omega(n^2)$[21, 32-33]. Depending on the data, complexity may actually be much greater.

Consider that at every step of the algorithm, the matrix representing the distance (or similarity, depending on the precise variant of the algorithm being used) between clusters must be searched, a new cluster must be formed, and then the distance matrix (or a portion of it) must be recalculated. In degenerate cases, where each step adds an original data element to a single cluster (note that Figure 1.1 is close to this situation) complexity may actually be $O(n^3)$. For most data sets, $O(n^2 \lg n)$ is probably a realistic estimate.

1.1.2 SiMCAL vs. Existing Methods

The origin of SiMCAL lies in the Jarvis-Patrick (JP) algorithm [16], which has been widely used in fields ranging from astronomy to chemoinformatics, but has so far not been common in bioinformatics applications. JP begins with two user-defined parameters: $k$ and $k_{\text{min}}$. For each data element, a list of that element’s $k$ nearest neighbors is calculated. Then the algorithm states that two data elements are in the same cluster if either of the following conditions applies:

- They are within each other’s lists of nearest neighbors.

- They have at least $k_{\text{min}}$ nearest neighbors in common.

The advantages of JP are threefold. First, because it allows chains of relationships between data elements — e.g., $a$ may be clustered with $b$, and $b$ clustered with $c$, so that $a$ and $c$ end up in the same cluster even if they are not obviously related to each other — it allows non-convex clusters which may more accurately
reflect patterns in the data. Second, it is largely non-parametric; it does not impose an arbitrary threshold on similarity between elements in order for them to be clustered together, but operates primarily by ranking, and is thus less sensitive to outliers than parametric methods. Third, and perhaps most important, it provides a method of determining the “natural” number of clusters within the data, rather than requiring a pre-selected figure as with k-means or an arbitrary cut as with hierarchical clustering.

However, it has disadvantages as well. The values of \( k \) and \( k_{\text{min}} \) are arbitrary and must be pre-selected; also arbitrary is the definition of what degree of similarity constitutes “neighborness.” Also, depending on the size of these values, it may be computationally expensive; because of the requirement to compare lists of neighbors, the algorithm is \( O(kn^2) \), probably less complex than hierarchical clustering but much more expensive than k-means. Finally, it provides only one level of resolution — the algorithm in its basic form provides no way of building hierarchical levels of clusters which may contain each other.\(^1\)

The idea of SiMCAL is to address these shortcomings, as well as to provide additional functionality not found in any of the algorithms so far examined. To meaningfully be labeled “SiMCAL,” an algorithm must possess the following attributes:

1. It must be *simple*, in that it has low computational complexity, preferably \( O(n^2) \) or less, and/or relatively few and lightweight computations are performed.

\(^1\)It is, of course, possible to provide multiple levels of resolution by iterating through possible values of \( k_{\text{min}} \); SiMCAL 1 uses a similar but not identical method, optimized for low computational expense.
formed at each step of the inner loop. Subjectively, the user should feel that
the algorithm is “fast” compared to other methods.

2. It must be multilevel, in that it provides a hierarchical relationship between
clusters, but with fewer levels than simple hierarchical clustering provides.
Therefore, the user can, to use a laboratory metaphor, “switch the objective”
to examine the results at a few well-defined levels of resolution.

3. It must provide linking between clusters, in that it yields information on the
relationship between different clusters at the same level in the hierarchy.

SiMCAL 1, 2, and 3 each fulfill these requirements. With regards to compu-
tational complexity, they are strictly $\Theta(n^2)$, in contrast to hierarchical clustering.
Also, their inner loop calculations are much more lightweight than those in hier-
archical clustering. Unlike k-means, however, algorithms in the SiMCAL family
do establish a hierarchical relationship between clusters. Finally, they share the
advantage of JP in that they show the “natural” number of clusters in the data
at each level in the hierarchy, while being less expensive than JP and offering the
unique advantage of providing information on the relationship between different
clusters at the same level.

The nature of this relationship varies depending on the type of data being
analyzed. In SiMCAL 1, the relationship is temporal. The user can then trace
the relationship through multiple clusters by navigating a directed graph. In
SiMCAL 2, the relationship is spatial, based on genes which are shared between
clusters, and the user can navigate an undirected graph. SiMCAL 3 offers both
temporal and spatial linking between clusters.
1.1.3 Project History

The original Fall 2002 version of the project, which used k-means clustering, was written using PostgreSQL[31] as the database server, PL/pgSQL[30] as the language for execution — data import, preprocessing, and clustering — and database interaction, and PHP[29] for the visualization and reporting interface. A Spring 2003 version, also based on k-means, was written using MySQL[20] as the database server, Perl[28] and Perl Data Language[27] for execution and database interaction.

Work on the SiMCAL 1 algorithm began in Fall 2003. The Summer 2004 version, which is identical in all but minor detail to the current version of SiMCAL 1, uses a command-line application written in Python[32] and Numerical Python[24], or “NumPy” (a set of numerical extensions to Python which provide similar functionality to dedicated numerical languages such as MATLAB[18] and Octave[25]) and MySQL as the DBMS. SiMCAL 2, developed in Fall 2004 and Spring 2005, and SiMCAL 3, developed in Summer 2005, also use NumPy and MySQL. The Web-based visualization and reporting interface remains in PHP, and Gnuplot[11] is used to produce high-quality gene expression plots which can then be displayed as part of this interface.

The use of these various open-source tools, all of which are free for non-commercial (i.e., academic and research) use, has been a powerful factor in making the project financially feasible, and in contributing to rapid development.

Each of the algorithms is designed for a slightly different type of data, which will be covered in detail in later chapters. In brief, SiMCAL 1 is designed for clustering time-series data, with the goal of finding clusters of genes with simi-
lar behavior patterns over time under experimental conditions and then finding succession relationships between these genes and the clusters which contain them. SiMCAL 2 is designed for analysis of single-point measurements of gene expression under both experimental and control conditions, in order to determine which genes are differentially expressed under experimental conditions, and finding spatial relationships between the clusters. SiMCAL 3 combines the two approaches, offering a means to find differential expression of genes in time-series data and then find succession relationships between them and their clusters.

Overall, although all the SiMCAL algorithms are considerably more complex than k-means, careful selection of language and platform as well as steady tuning of the algorithms has resulted in better than an order of magnitude performance improvement over the original version — from ten to fifteen hours for all operations (data import, preprocessing, and analysis) in the original version, to under an hour today, on a midrange laptop. Performance on a dedicated server should be excellent.

1.2 Biological Background

Both the phosphatidylinerse receptor (PSR) and cystic fibrosis (CF) are areas of intensive research. PSR appears to play a vital role in mediation of inflammatory response, and understanding the mechanism by which PSR fulfills this role may be important to research on the mechanisms of many diseases, including CF. CF is a serious and widespread disease which results in significant morbidity and mortality, and is presumed to involve large portions of the human genome; thus being able to determine which genes are differentially expressed in CF patients may lead to important advances in diagnosis and treatment.
Both PSR and CF are covered in this section. PSR is covered in detail because it is a newer topic and not as widely known. CF is covered in brief; the information given here is summarized from [3, 2366-2371] unless otherwise cited.

1.2.1 The Phosphatidylserine Receptor (PSR)

Normal cell death, or \textit{apoptosis}, requires regular housekeeping activity within all multicellular organisms. In vertebrates, this activity is performed by macrophages, which must be able to distinguish between three classes of cells: healthy cells, which the macrophages must not consume; apoptotic cells, which must be consumed without triggering an inflammatory response; and cells undergoing \textit{lysis} — death from some external cause such as injury or disease, which must also be consumed, but which should trigger an inflammatory response[19].

In invertebrates such as the ubiquitous \textit{C. elegans}, less specialized cells perform housekeeping activities with respect to both apoptotic and lytic neighbors[8], but the same challenges are presented. Since cells are constantly undergoing apoptosis throughout the bodies of all multicellular organisms, the mechanism for preventing inflammatory response must work perfectly or disease will result. In humans, serious diseases such as CF, lupus, and some forms of diabetes may be at least partly the result of a breakdown of this mechanism[37].

1.2.1.1 Significance of PS and PSR

\textit{Phosphatidylserine} (PS) is a phospholipid generally found on the cytosolic surface of the cell membrane in most animal cells. Apoptotic cells appear to shift PS rapidly to the exterior of the membrane, at least partly as a signal for phagocytosis. However, it is far from the only signal for this process. What makes PS interesting?
The answer is the effect PS has on the macrophage. The PS receptor, or PSR, is believed to be “a crucial molecular switch” in the control of immune response[15]. Specifically, when PS is expressed in apoptotic cells, it appears to reduce inflammation and other aspects of the immune response by engaging the PSR in macrophages. Cells undergoing lysis do not show this behavior, allowing the full range of immune response, including inflammation, to come into play.

The fact that PSR is a highly conserved protein, present and functional in species from *C. elegans* to *H. sapiens*, indicates that it plays an important role.² In some physiological systems, apoptosis is such a frequent event that any inflammatory response it might produce must be downregulated to the point of elimination. For example, the entire pool of neutrophils (a type of white blood cell) in the body turns over about 2.5 times per day[15]. Any significant inflammatory response to such a high rate of apoptosis would (and, in diseases such as systemic vasculitis, does) have disastrous physiological consequences.

### 1.2.1.2 The PS/PSR Mechanism

Fadok, Henson et al. propose a “tether and tickle” mechanism for the action of the PS-PSR complex. A variety of adhesion ligands on the surface of the apoptotic cell induce the initial attachment of the macrophage — the “tether.” But the “tickle” provided by the apoptotic cell’s PS to the macrophage’s PSR is what signals the macrophage to both engulf the cell and to mediate immune response.

²It is generally accepted that highly conserved genes, and therefore the proteins they express, are highly functional in the species across which they are conserved. The assumption is that important sequences which undergo significant mutation will have dramatic, and generally unfortunate, consequences for the evolutionary fitness of their carriers, while less important sequences are more likely to change randomly over time without having such effects.
PSR antibodies may cause the macrophage to down-regulate inflammation by inducing the secretion of TGF-β[19].

Mature macrophages possess a wide variety of recognition molecules on their surfaces to bind to apoptotic cells. Many of these molecules, such as the β-integrins, have very low specificity — essentially, they act as signals for the macrophage to bind to any of its possible targets, including lytic cells which normally create an inflammatory response[8]. This, of course, is the tether. The low specificity of these molecules allows the macrophage to bind to both apoptotic and lytic targets with ease, but it also means that they cannot act as regulators for the immune response. That is left for the tickle, a more specific molecular signal: the interaction of apoptotic cell PS with macrophage PSR.

Phosphatidylserine is found on the surface of all apoptotic cells[15]. In healthy cells, PS is confined to the cytosolic (inner) leaflet of the membrane bilayer, except for transient expressions on the exoplasmic (outer) leaflet which are quickly corrected. During apoptosis, phospholipid scramblases allow chaotic movement of phospholipids in both directions across the membrane. (Low-level activity of these scramblases in healthy cells probably accounts for the transient expression of PS on the cell surface.) PS which approaches the cell surface in healthy cells is relocated inside the membrane by aminophospholipid translocases (APLTs). In contrast, APLTs seem to inactivate or be overwhelmed during apoptosis.

The observation that close homologues of PSR are to be found in *C. elegans* and *D. melanogaster*, organisms which lack macrophages\(^3\) goes hand-in-hand with

\(^3\) *D. melanogaster* possesses specialized immune cells called hemophages, which function similarly to macrophages in vertebrates.
the fact that most tissue cells, not just dedicated phagocytes, can ingest apoptotic cells[8]. This is an elegant mechanism for dealing with the immunological problem presented by apoptosis.

Tethered cells (macrophages or general tissue cells) which are stimulated by PSR increase their uptake not only of the PS expressing cell, but of other cells to which they are tethered[8]. Furthermore, blocking the effect of PSR greatly reduces the uptake of apoptotic cells in general. Both the tether and the tickle are necessary for apoptotic cell uptake, but neither by itself is sufficient. Together, they provide a powerful and strikingly efficient mechanism for disposal of apoptotic cells.

PSR seems to have far-reaching effects on various aspects of possible inflammation. PSR on immature dendritic cells appears to inhibit maturation of these cells, which in turn prevents these cells from processing and presenting various antigens which might cause inflammation. Since mature dendritic cells are required for processing of most antigens in order for the antigens to be recognized by the rest of the immune system, this effect of PSR constitutes a powerful immunosuppressive and thus anti-inflammatory effect[15].

1.2.1.3 Avenues of PS/PSR Research

One major question about the PSR effect is why lytic cells, which also contain PS which normally resides on the cytosolic leaflet of the cell membrane but can be expressed on the exoplasmic face during cell death, do not engage PSR. Obviously this would be a physiologically inappropriate response; the consequences of suppressing inflammation in the case of lysis could be just as severe as those of allowing inflammation in the case of apoptosis, because inflammation is a cru-
cial component of the immune response. (The major functions of inflammation in immune response are to attract and activate immune system components such as B and T lymphocytes, to begin tissue repair and remodeling, and to contain pathogens at the inflamed site rather than allowing them to spread throughout the body[36, 440-443].) Although the mechanism by which lytic cells avoid engaging PSR is not yet known, there are at least two proposed mechanisms:

First, cytoplasmic proteins such as annexins may bind PS and prevent it from engaging PSR. Lysis is a violent process compared to apoptosis, and interactions between the cell membrane and the cytoplasm during lysis might provide an opportunity for such binding.

Second, proteases which are released only during lysis might cleave PSR in the engulfing cells (macrophages in vertebrates, general tissue cells in invertebrates.) PSR is highly susceptible to protease cleavage, and lytic cells express proteases to a degree not found in cells in any other condition. Less dramatic protease cleavage of PSR might also be involved in allowing immature dendrites to mature in cases where this response is appropriate.

Clinically, high levels of serine proteases such as neutrophil elastase are often found in patients suffering from inflammatory disorders, or disorders which have a significant inflammatory component, such as CF and bronchiectasis[37]. It may be that these proteases would make useful targets for drug therapy which, by reducing the protease levels, would allow more PSR regulation of inflammatory response.

Because of their cell walls, bacterial and fungal pathogens cannot express PS on their surfaces, and therefore cannot engage PSR. On the other hand, tumor
cells can and do express PS — one of the main reasons why the immune system is much less effective against cancer than against infection.

1.2.2 Cystic Fibrosis (CF)

CF is a hereditary disease which occurs in about 1 of every 5,000 births. It is one of the most common fatal genetic diseases, and the median age of death for CF patients is only 31 years. Although primarily thought of as a respiratory disease, it can affect many major body systems: symptoms include not only respiratory insufficiency and increased sputum production resulting in coughing and difficulty sleeping, but also growth deficiency, heart disease, increased vulnerability to many infectious diseases, and pancreatic insufficiency leading to insulin-dependent diabetes. Most exocrine glands (those which secrete substances which are transported outside the body or to body cavities, via the skin or mucous membranes) are also affected by the disease, often leading to digestive difficulties and infertility among other problems.

The cause of CF is believed to be a mutation on a single gene on human chromosome 7, which is carried asymptptomatically as a recessive trait by 2-3% of the population. However, the disease’s wide-ranging effects on the body make it reasonable to expect that many genes in CF patients will be either up- or down-regulated, making patients with the disease (and, of course, healthy control subjects) fruitful subjects for genetic analysis, including microarray experiments. Complicating the genetic analysis of the disease is the fact that, although the disease is probably caused by a mutation on a single gene, there are 600+ distinct mutations to this gene which may be able to cause the disease to a greater or lesser degree of severity.
The various mutations associated with CF all cause sequence and structural changes in the cystic fibrosis transmembrane regulator (CFTR) protein, which appears to be part of a chlorine (Cl) and sodium (Na) transport channel across epithelial membranes (membranes lining the surface of the body and body cavities.) This channel is regulated by cyclic adenosine monophosphate (cAMP), a substance which plays a vital role in metabolism[38, 442]. Thus genes which might be expected to be up- or down-regulated in CF patients include those associated with CFTR, with Cl and Na transport, with cAMP synthesis and breakdown, with metabolism in general, and with the various organs and body systems affected by the disease — in short, a significant portion of the entire human genome.

1.3 Notations and Definitions

It is useful here to define some notations and terms used throughout the following sections. Equations in this section which are not numbered may be found in [6, 113-117, 216-218]. Numbered equations are original, unless otherwise cited.

Given two vectors \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \) and \( \mathbf{y} = (y_1, y_2, \ldots, y_n) \), and a scalar \( z \); and for arbitrary functions \( f(\mathbf{x}) \) such as \( x^2 \), \( g(\mathbf{x}, \mathbf{y}) \) such as \( xy \), or \( h(\mathbf{x}, z) \) such as \( x + z \), unless otherwise specified:

\[
f(\mathbf{x}) = (f(x_1), f(x_2), \ldots, f(x_n)); \tag{1.1}
\]
\[
g(\mathbf{x}, \mathbf{y}) = (g(x_1, y_1), g(x_2, y_2), \ldots, g(x_n, y_n)); \tag{1.2}
\]
\[
h(\mathbf{x}) = (h(x_1, z), h(x_2, z), \ldots, h(x_n, z)). \tag{1.3}
\]
The mean or expected value of $x$ is:

$$\mu_x = \frac{1}{n} \sum_{i=1}^{n} x_i.$$  

The variance and standard deviation of $x$ are:

$$V(x) = \mu(x-\mu_x)^2 = \mu_x^2 - \mu_x^2;$$  

$$\sigma_x = \sqrt{V(x)}.$$  

The covariance of $x$ and $y$ is:

$$\text{Cov}(x, y) = \mu(x-\mu_x)(y-\mu_y) = \mu_{xy} - \mu_x\mu_y.$$  

The correlation coefficient $[6, 218-220]$ of $x$ and $y$ is:

$$\rho_{x,y} = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y}. \quad (1.4)$$  

If $X$ is a matrix such that $X = (x_1, x_2, \ldots, x_m)$ but $y$ is a vector as above:

$$\rho_{X,Y} = (\rho_{x_1,y}, \rho_{x_2,y}, \ldots, \rho_{x_m,y}). \quad (1.5)$$  

If $X$ and $Y$ are both matrices such that $X = (x_1, x_2, \ldots, x_m)$ and $Y = (y_1, y_2, \ldots, y_n)$, then the correlation matrix is:

$$\rho_{X,Y} = \begin{pmatrix} \rho_{x_1,y_1} & \rho_{x_1,y_2} & \cdots & \rho_{x_1,y_n} \\ \rho_{x_2,y_1} & \rho_{x_2,y_2} & \cdots & \rho_{x_2,y_n} \\ \cdots & \cdots & \cdots & \cdots \\ \rho_{x_m,y_1} & \rho_{x_m,y_2} & \cdots & \rho_{x_m,y_n} \end{pmatrix}. \quad (1.6)$$
The weighted mean of \( x \) and \( y \) is the mean of \( x \) weighted by the values of \( y \). In other words, if \( y = (1, 2, 1) \), then in the calculation of the mean, \( x_2 \) has a weight equal to the combined weights of \( x_1 \) and \( x_3 \), which have weights equal to each other. The definition of the function depends on the dimension of its arguments. For \( x = (x_1, x_2, \ldots, x_n) \) and \( y = (y_1, y_2, \ldots, y_n) \):

\[
W(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} y_i}. \tag{1.7}
\]

If \( X \) is a matrix such that \( X = (x_1, x_2, \ldots, x_m) \) but \( y \) is a vector as above:

\[
W(X, y) = (W(x_1, y), W(x_2, y), \ldots, W(x_m, y)). \tag{1.8}
\]

If \( X \) and \( Y \) are both matrices such that \( X = (x_1, x_2, \ldots, x_m) \) and \( Y = (y_1, y_2, \ldots, y_m) \), then:

\[
W(X, Y) = (W(x_1, y_1), W(x_2, y_2), \ldots, W(x_m, y_m)). \tag{1.9}
\]

Let \( \max_0(x) \) and \( \max_0(X) \) be defined as:

\[
\max_0(x) = (\max(x_1, 0), \max(x_2, 0), \ldots, \max(x_n, 0)). \tag{1.10}
\]

\[
\max_0(X) = (\max_0(x_1), \max_0(x_2), \ldots, \max_0(x_m)). \tag{1.11}
\]

If \( X \) is \( m \times n \), then the transpose of \( X \) \([9, 17]\) is \( X^T \) such that:

\[
X_{j,i}^T = X_{i,j} \forall i \in [1, m], j \in [1, n] \tag{1.12}
\]
The identity matrix of size \( n \) \([9, 83]\) is the \( n \times n \) matrix \( I_n \) such that:

\[
I_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad \forall i, j \in [1, n]. \tag{1.13}
\]

Minkowski distances are widely used distance measures of the family:

\[
\Delta_p(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}.
\]

The best-known Minkowski distance is probably the Euclidean\([4, 377]\):

\[
\Delta_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{1.14}
\]

Another very useful Minkowski distance is the simplest possible case, the city block distance\([4, 377]\):

\[
\Delta_1(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{1.15}
\]

Given two sets \( A \) and \( B \), and that for any set \( S = \{s_1, s_2, \ldots, s_n\} \), the cardinality of \( S \) is given by \( |S| = n \), let the congruence of \( A \) and \( B \) be defined as:

\[
\text{Con}(A, B) = \begin{cases} 0 & \text{if } A = \emptyset \text{ or } B = \emptyset \\ \frac{|A \cap B|^2}{|A||B|} & \text{otherwise} \end{cases} \tag{1.16}
\]

It can be seen that \( \text{Con}(A, B) \) is a dimensionless measure of the degree to which \( A \) and \( B \) intersect: \( \text{Con}(A, B) \in [0, 1] \forall A, B \), and \( A = B \Rightarrow \text{Con}(A, B) = 1 \), while \( A \cap B = \emptyset \Rightarrow \text{Con}(A, B) = 0 \).
2. SiMCAL 1

SiMCAL 1, the original SiMCAL algorithm, is designed primarily to cluster genes based on their behavior under experimental conditions over time[7]. A secondary goal is to determine which genes and clusters accurately predict the behavior of others. This chapter covers SiMCAL 1 in detail, and introduces a number of concepts which will also be used in subsequent chapters.

2.1 Data and Preprocessing

To ensure that the data were accurate and replicable, three separate and identical experiments (replicants) were performed. In each, microarrays containing 12,488 mouse genes known to be active in mouse macrophages, with each gene being represented by between 5 and 30 RNA probes, were exposed to a single concentration of monoclonal PSR antibody over time. Fluorescent hybridization measurements were taken for RNA at 30 minutes, 2 hours, and 8 hours 30 minutes. Thus, using the standard notation by which \( n \) denotes the number of variables and \( m \) denotes the number of observations, \( n = 12488 \) and \( m = 3 \).

Identical measurements were taken of control samples which were treated with an isotype, i.e., another IgM antibody with better-known effects. Another control sample, of which only one measurement was taken, was not treated. Therefore, seven distinct data points were generated for each probe. These steps were performed using Affymetrix (“Affy”) microarray equipment and the Affy chip MG-U74Av2, a standard chip containing mouse genes.

Preprocessing is of fundamental importance in any data mining operation. The raw data, although interesting, do not yet contain usable information. The
primary goals of preprocessing in this case are to eliminate or reduce unreliable data, combine separate data points in a way that takes all measurements into account, and to normalize the data in such a way that clustering and linking will produce meaningful results.

2.1.1 Raw Measurements and Reliability

For each “raw” gene in each replicant, there exists a set of measurements, based on one measurement for the unexposed control, and time series measurements for the isotype-exposed genes, iso, and the PSR-exposed genes, psr. In the time series measurements, \( psr_t \) and \( iso_t \) denote the values measured at time point \( t \). For the current data, \( t = 1, t = 2, \) and \( t = 3 \) indicate exposure times of \( \frac{1}{2} \) hour, 2 hours, and \( 8\frac{1}{2} \) hours respectively. Therefore, overall:

\[
\text{measurements}_{\text{gene}\in\text{replicant}} = \{\text{control}\} \bigcup_{t=1}^{3} \{iso_t, psr_t\}. \tag{2.1}
\]

Each of these measurements has an associated p-value, and a lower p-value indicates higher reliability placed on the measurement\([1]\). A reliability can be calculated for any gene in a replicant based on the average p-values. First, let \( \varepsilon = \log_{0.95} 0.05 \approx 58.4, \) and then reliability is:

\[
\text{rel}_{\text{gene}\in\text{replicant}} = \min(1 - (pvalues_{\text{gene}\in\text{replicant}}))^\varepsilon. \tag{2.2}
\]

In other words, for any gene in any replicant, that gene’s reliability is the minimum of the reliability for any measurement associated with it. One bad measurement in an otherwise good time series could badly throw off all subsequent calculations; by this method, the algorithm is aware of such measurements, and will either discard them or assign them less weight in calculations, as described
in following sections. Eqn. 2.2 also ensures \( rel \in [0, 1] \), providing a simple and consistent basis for calculation and interpretation.

The definition of \( \varepsilon \) deserves some discussion. Affy’s own evaluation of measurements reliability marks measurements having a p-value significantly above 0.05 as “absent,” those having a p-value around 0.05 as “marginal,” and those having a p-value significantly less than 0.05 as “present.” The definition of \( \varepsilon = \log_{0.95} 0.05 \) means that a “marginal” gene having a maximum p-value of 0.95 will have a reliability of 0.05, or “5%.” This is the minimum reliability used in any subsequent calculation and leads to very low — but not zero — weighting for measurements which have a p-value only slightly less than or equal to 0.05.

This choice is arbitrary; it would be possible to let \( \varepsilon = 1 \), or, at the opposite extreme, a very large value such as \( \varepsilon = 1000 \). In any case, \( \varepsilon > 1 \) serves the purpose of “concentrating” the more reliable values\[4, 81\]: the higher its magnitude, the more emphasis is given to those expressions with lower p-values.

### 2.1.2 Combination and Normalization

Before calculating normalized values, replicants with \( rel < 0.05 \) are discarded. This ensures that the analysis will proceed only with reliable data. Now, a consistent basis is required for comparing expression levels between genes for which reliable measurements exist.

Unfortunately, there no meaningful way to compare the raw expression levels to each other for different genes; it cannot be said that if one gene at one time point expresses at 5000 according to the Affy reading, and another at the same time point expresses at 10000, that the second gene is expressing twice as much of the relevant protein as the first. However, it can be said that if a single gene
expresses at 5000 at one time point and at 10000 at the next, that its expression level has doubled[1].

Therefore, a measurement of the relative expression levels of the PSR-exposed data points relative to the isotype-exposed data points and the untreated control data point is defined in Eqn. 2.3. Note that \( \lg x = \log_2 x \), that is, the base-2 logarithm of \( x \).

\[
exp_t = \lg \frac{\text{psr}_t^2}{\text{control} \times \text{iso}_t}, \ t \in \{1, 2, 3\}.
\]  

(2.3)

Now let the expression level for the untreated control be \( exp_0 = 0 \), and:

\[
\exp_{\text{gene}\_\text{replicant}} = (exp_0, exp_1, exp_2, exp_3).
\]

(2.4)

Each expression level has been calculated to take into account the change between the control expression level and the PSR-exposed expression level, and just as importantly, the changes between the PSR-exposed expression level and the isotype-exposed expression level. In other words, the normalized expression level should reflect the difference PSR specifically makes in expression, as well the difference PSR as an IgM antibody makes in expression. Consider two possible cases:

\[
\frac{\text{psr}_t^2}{\text{control} \times \text{iso}_t} = \begin{cases} 
9 & \text{if } (\text{control}, \text{iso}_t, \text{psr}_t) = (1000, 1000, 3000); \\
3 & \text{if } (\text{control}, \text{iso}_t, \text{psr}_t) = (1000, 3000, 3000).
\end{cases}
\]

In the first case, PSR specifically is clearly responsible for up-regulating the gene under examination, while in the second case, it appears to be the presence of
IgM antibody in general which is responsible for the up-regulation. Appropriately, therefore, the first case has a much higher normalized value.

A logarithmic scale for normalized expression levels is used for two reasons, both relating to revealing small differences in the expression levels. First, the values of \( \frac{psr_i^2}{(control \times iso_i)} \) cover an enormous range, from roughly 0.00003 to 3500, and clearly any reporting mechanism which can cover such a range would lack the precision to distinguish between a subtle but important difference such as that between 3 and 9. Second, such a reporting mechanism would tend to overemphasize up-regulation and underemphasize down-regulation, since the magnitude of the values for up-regulated genes is so much greater.

Finally, 2 is used as the base of the logarithm, rather than some other common choice such as \( e \) or 10, because it makes interpretation straightforward: a change of 1 in the reported value means a twofold change in expression. The use of base-2 log ratios has become standard in processing raw microarray expression values; some recent examples are to be found in [12], [13], and [23]. The particular ratio used in Eqn. 2.3 is original.

### 2.1.3 Gene Expression Calculation

At this point, reliability and expression values exist for each gene in each replicant. The reliability score is used as a weighting factor in the averaging. In addition to the expression vector \( \text{exp} \) and the reliability \( \text{rel} \), it is also useful to calculate variability, \( \text{var} \), which is an overall measure of how much the gene reacts to the experimental conditions. For each gene:

\[
\text{exp}_{\text{gene}} = W_{\text{replicants}}(\text{exp}_{\text{gene}\in\text{replicant}}, \text{rel}_{\text{gene}\in\text{replicant}});
\] (2.5)
\[ \text{rel}_{\text{gene}} = \mu_{\text{rel}_{\text{gene}}} \in \text{replicant}; \quad (2.6) \]

\[ \text{var}_{\text{gene}} = \frac{1}{3} \sum_{t=1}^{3} \text{abs}(\exp_{\text{gene}}). \quad (2.7) \]

Genes are assigned Id numbers, ordered from least variable (Id 0) to most variable (with the current data set, for all genes that made it through preprocessing, Id 5285). They may then be considered as an array, and individual genes denoted by their position in the array — for example, the gene with the Id number 2201 is \( G_{2201} \).

### 2.2 Clustering and Linking

Now clustering can begin. This section describes the distance measures used and how clusters are formed, associations are mapped, and genes and clusters are linked from this information.

#### 2.2.1 Distance Measures and Matrices

The similarity measure, for any two genes, used throughout the clustering and linking calculations is the correlation coefficient \( \rho \) as defined in Eqn. 1.4. This is a general measure of the degree of linear dependence between two vectors — i.e., the degree to which they tend to rise or fall together\([6, 218-220]\).

This measure, often referred to as the Pearson correlation coefficient, has become a canonical measure of similarity (or, in the inverse, difference) between gene expression patterns in a wide variety of applications; some of many recent examples include \([13], [17], [35], \) and \([39]\). More complex time-series measures, such as Box-Jenkins, generally require a larger number of data points than those generated in this experiment, and also often have a requirement that the data be
evenly spaced; in contrast, $\rho$ is robust for multidimensional data of almost any dimensionality and interval.

Compared to other dependence measures, $\rho$ has the useful quality that it is dimensionless: $\rho_{xy} \in [-1, 1]$ for all vectors of equal length $x$ and $y$. Interpretation is straightforward:

- $\rho_{xy} = 0$ indicates that $x$ and $y$ are linearly independent.
- $\rho_{xy} \to 1$ indicates increasing positive linear dependence; as $x$ rises or falls, so does $y$. Similarly, $\rho_{xy} \to -1$ indicates increasing negative linear dependence; as $x$ rises, $y$ falls, and vice versa.
- $|\rho_{xy}| \geq 0.8$ indicates a strong linear dependence between $x$ and $y$, $0.5 < |\rho_{xy}| < 0.8$ indicates moderate dependence, and $0 < |\rho_{xy}| \leq 0.5$ indicates weak dependence.

Why use a dependence measure as opposed to the common Minkowski distance measures (Ch. 1.3) such as the Euclidean? The answer is that clustering is based here on similarity in the changes in expression patterns under experimental conditions, and a dependence measure such as $\rho$ provides a better measure of the relationship between different genes’ up- or down-regulation over time than does a Minkowski distance. Consider the following three vectors (see Fig. 2.1) which represent possible values of $\text{exp}_{\text{gene}}$:

- $x = (0, 0.5, 1, 1.5)$;
- $y = (0, 2, 4, 5)$;
- $z = (0, 0, -0.5, -0.5)$.
Clearly $y$ and $z$ are closer to each other by any Minkowski measure than either is to $x$; just as clearly, however, the responses of $x$ and $y$ under experimental conditions are more similar, since both are steadily upregulated while $z$ is downregulated. Comparing $1 - \rho$ to $\Delta_2$ (so as to cast both values as measures of distance, rather than similarity) shows that the choice of $\rho$ as a similarity measure reflects this.

As Table 2.1 shows, clustering these genes based on $\Delta_2$ would create the clusters $\{x, z\}$ and $\{y\}$, but clustering based on $\rho$ creates $\{x, y\}$ and $\{z\}$. Given the actual reactions of the genes to experimental conditions, $\rho$ is a much more
Table 2.1: \( 1 - \rho \) vs. \( \Delta_2 \).

<table>
<thead>
<tr>
<th></th>
<th>( 1 - \rho ) : ( \Delta_2 )</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.00:0.00</td>
<td>0.01:4.85</td>
<td>1.89:2.55</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>0.01:4.85</td>
<td>0.00:0.00</td>
<td>1.91:7.38</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>1.89:2.55</td>
<td>1.91:7.38</td>
<td>0.00:0.00</td>
<td></td>
</tr>
</tbody>
</table>

reasonable measure.

The use of \( \rho \) is the reason that \( \exp_0 = 0 \). Recall that each gene is starting from the “natural” (untreated) state, and changes in expression level are measured from this baseline. If the expression vector were simply \( \exp_{\text{gene}} = (\exp_1, \exp_2, \exp_3) \), this could lead to false correlations.

Suppose \( x = (-1, -3, -2) \) and \( y = (2, 0, 1) \). (See Fig. 2.2.) Without \( \exp_0 = 0 \), \( \rho_{x,y} = 1 \), which is clearly inaccurate, since in fact \( x \) is downregulated under experimental conditions while \( y \) is upregulated. Setting \( x = (0, -1, -3, -2) \) and \( y = (0, 2, 0, 1) \) gives a much more accurate value of \( \rho_{x,y} = 0.13 \).

Therefore, the initial step in clustering is to form a correlation matrix for all genes under consideration. This is the most complex step in the entire algorithm, but is strictly \( O(n^2) \), and is still computationally inexpensive because calculating \( \rho \) is quite fast. Thus SiMCAL 1 as a whole is \( O(n^2) \), and the first SiMCAL condition is fulfilled.

For the \( n \) genes \( g \in [1, n] \) and \( \exp_g = (\exp_{g,0}, \exp_{g,1}, \exp_{g,2}, \exp_{g,3}) \), let \( \exp_{g,\text{early}} = (\exp_{g,0}, \exp_{g,1}, \exp_{g,2}) \) and \( \exp_{g,\text{late}} = (\exp_{g,1}, \exp_{g,2}, \exp_{g,3}) \). Thus there are three expression matrices:
Figure 2.2: Expression levels of x and y.

\[
\text{Exp} = (\exp_1, \exp_2, \ldots, \exp_n);
\]

\[
\text{Exp}^{\text{early}} = (\exp_{1}^{\text{early}}, \exp_{2}^{\text{early}}, \ldots, \exp_{n}^{\text{early}}); \tag{2.8}
\]

\[
\text{Exp}^{\text{late}} = (\exp_{1}^{\text{late}}, \exp_{2}^{\text{late}}, \ldots, \exp_{n}^{\text{late}}).
\]

These expression matrices are used to obtain the similarity matrices which will be used in clustering. First, the “raw” matrices are:
Neighbors = \max_0(\rho_{\text{Exp,Exp}});
Leaders = \rho_{\text{Exp}^{\text{late}},\text{Exp}^{\text{early}}};
Followers = \text{Leaders}^T. \tag{2.9}

The derivation of Neighbors is obvious. Leaders and Followers are somewhat deceptively named, but once they are used to obtain ranks based on the values of their elements (as described in the next section) they do indeed provide a measurement of the degree to which \text{Exp}^{\text{early}} predicts \text{Exp}^{\text{late}}. In other words, the maximum value in the \(i\)th row of Leaders is the index of the gene which best predicts gene \(i\): its “Leader”. Similarly, the maximum value in the \(i\)th row of Followers is the index of the gene which gene \(i\) best predicts: its “Follower”.

However, the matrices in their current form are not adequate for clustering to proceed, and must be adjusted. First, Neighbors, used in clustering, is subtracted from both Leaders and Followers, used in linking:

\[
\begin{align*}
\text{Leaders} &= \text{Leaders} - \text{Neighbors}; \\
\text{Followers} &= \text{Followers} - \text{Neighbors}. \tag{2.10}
\end{align*}
\]

This step serves to prevent false leader-follower (“succession”) associations between genes whose expression levels track each other very closely under experimental conditions. Such genes should be clustered together using Neighbors, but they should not be linked using Leaders or Followers.

To illustrate this, suppose that for two genes \(x\) and \(y\), and some constant \(c > 0\), \(y \approx cx\). In other words, \(x\) and \(y\) approach complete positive linear dependence. Then “\(x\) leads \(y\)” and “\(y\) leads \(x\)” would be equally accurate statements. Since the purpose of the succession analysis is the elucidation of biological pathways,
based on the assumption that a genes’ leaders precede it in these pathways and its followers succeed it, this would not be very helpful. Thus the subtraction step, which ensures that the more likely two genes are to be clustered, the less likely they are to be linked.

Finally, genes should not be clustered or linked with themselves; this is ensured by by reducing their reported associations to below $\min(\rho)$. To accomplish this, first let the matrix $\text{Mask} = 2I_n$ (see Eqn. 1.13) where $n$ is the number of genes. Then $\text{Mask}$ is subtracted from all three similarity matrices:

\[
\begin{align*}
\text{Neighbors} &= \text{Neighbors} - \text{Mask}; \\
\text{Leaders} &= \text{Leaders} - \text{Mask}; \\
\text{Followers} &= \text{Followers} - \text{Mask}. 
\end{align*}
\] (2.11)

### 2.2.2 Ranks and Cluster Formation

The correlation matrices are now used to obtain rankings of association. Define the rank of a gene’s relationship to another gene as its ranking in the appropriate correlation matrix — that is, rank 1 corresponds to the index of the largest argument in a given row of the correlation matrix, rank 2 corresponds to the index of the second-largest argument, and so on.

Let $j = \text{argmax}(\text{Neighbors}_i)$. Then gene $\mathbf{x}_j$ is the rank 1 neighbor of $\mathbf{x}_i$. (Note that $i \neq j$ because of the mask subtraction step.) The rank 2 neighbor of $\mathbf{x}_i$ is $\mathbf{x}_k$ where $k = \text{argmax}(\text{Neighbors}_i) \mid k \neq j$; the rank 3 neighbor of $\mathbf{x}_i$ is $\mathbf{x}_l$ where $l = \text{argmax}(\text{Neighbors}_i) \mid l \neq j \land l \neq k$; etc. The calculations of leader and follower ranks proceed in similar fashion.

As implemented, this calculation is efficiently performed on all three similarity matrices ($\text{Neighbors}$, $\text{Leaders}$, and $\text{Followers}$) in their entirety, in one step,
using the NumPy argsort() function, which produces a sorted list of indices of a
matrix or vector, as shown below in Python sample code:

```python
a = array([1, 3, 2, 4])
b = argsort(a)
# b = array([0, 2, 1, 3])
```

In this example, \( a_0 \) is the rank 1 member of \( a \), \( a_2 \) is the rank 2 member, \( a_1 \) is the rank 3 member, and \( a_3 \) is the rank 4 member.

The fliplr() function simply flips a matrix or vector from left to right, which makes it easier to extract rank information in the order the algorithm demands.

Thus the code for finding ranked associations is:

```python
Neighbors = fliplr(argsort(Neighbors))
Leaders = fliplr(argsort(Leaders))
Followers = fliplr(argsort(Followers))
```

Now clustering can proceed. Clusters are ranked just as neighbors, leaders, and followers are; a cluster of rank 1 contains all genes which have each other as neighbors of rank 1, and so on through higher ranks until only one “supercluster” exists. In other words, the condition for clustering is that for any rank \( R > 0 \), a gene is in the same cluster of rank \( R \) as other genes which it has as neighbors of rank \( r \leq R \). Thus at rank 1, a gene is in the same cluster as its neighbors of rank 1 (and those genes’ neighbors of rank 1, and so on); at rank 2, it is in the same cluster as its neighbors of rank 1 or 2, etc.

This goal is achieved by starting, at rank 1, with each gene in its own “rank 0” cluster, then finding its neighbor of rank 1, and placing that gene, along with
any other genes which may be in its cluster, in the same cluster as the first gene. As the algorithm proceeds, of course, the clusters grow rapidly. Note that there is no requirement that genes be directly related to each other to be placed in the same cluster — this fulfills the requirement, inherited from JP, that the algorithm should be able to produce non-convex clusters. Instead, the algorithm navigates along chains of close association through the data. If \( x \) has \( y \) as its nearest neighbor, and \( y \) has \( z \) as its nearest neighbor, then \( x \) and \( z \) will end up in the same cluster of rank 1.

Clusters of higher rank are produced by reusing existing data structures. At rank 2, the genes are already clustered with their neighbors of rank 1, so there is no need to repeat the entire process with each gene’s entire list of neighbors. Instead, the existing rank 1 clusters are agglomerated into rank 2 clusters, and so on through higher ranks. This process allows a hierarchy of clusters of different ranks, expressed as parent and child relationships. Let \( A \) and \( B \) be clusters of rank \( r \), and \( C \) be a cluster of rank \( r + 1 \). If \( x \in A \), \( y \in B \), and \( x, y \in C \), then \( C \) is the parent of \( A \) and \( B \), and \( A \) and \( B \) are children of \( C \).

The following Python code from the clustering control file shows the core of the clustering procedure, at any rank:

```python
# cycle through genes
for GenId in xrange(NumOfGenes):
    # get ID’s into easy-to-use forms
    i = GeneId
    j = GeneNeighbors[rank-1, i]
    # cluster assignment
```
if GeneClusterIndexes[j] != GeneClusterIndexes[i]:
    # merge higher-ID cluster with lower-ID cluster
    localGCI = array([GeneClusterIndexes[i],
                      GeneClusterIndexes[j]])
    live = min(localGCI)
    dead = max(localGCI)
    # update GeneClusterIndexes to reflect merge
    for id in ClusterGenes[dead]:
        GeneClusterIndexes[id] = GeneClusterIndexes[live]
    # merge old cluster into new
    ClusterGenes[live] += ClusterGenes[dead]
    # empty out old cluster
    ClusterGenes[dead] = []
    # else: already in the same cluster

Clustering ends when the algorithm reaches a rank such that only one cluster exists. The algorithm could, of course, continue operations up to rank \( n - 1 \) for \( n \) genes. As a practical matter, doing so would serve no useful purpose. In fact, to speed up overall operations and reduce the amount of data storage required, the constant MAX_RANK is defined; no neighbor, leader, or follower associations beyond this rank are calculated or stored. With the current data set, clustering terminates at rank 4, and MAX_RANK is (generously) set equal to 5. Determining the proper value of MAX_RANK is a sort of preprocessing step, based on early analysis of the data in question, and of course the value may be changed for new data.
Cluster centers are calculated, although they exist only for reporting and visualization purposes rather than any further algorithmic uses. Like a gene, a cluster has the expression vector $\text{exp}_{\text{cluster}} = (\exp_0, \exp_1, \exp_2, \exp_3)$. For a given cluster $C$ with $m$ genes as members, let $G$ be the matrix of the expression vectors of the genes in the cluster:

$$G = (x_1, x_2, \ldots, x_m) \mid x_1, x_2, \ldots, x_m \in C.$$  

Also let $\text{rel}$ be the vector of the reliabilities of those genes:

$$\text{rel} = (\text{rel}_1, \text{rel}_2, \ldots, \text{rel}_m).$$

Then the location of the cluster center is:

$$\text{exp}_C = W(G, \text{rel}).$$  

Like genes, clusters are given Id numbers, starting at 0 within each rank and going up to the last cluster in the rank. They are not ordered by variability or any other inherent feature, although practically speaking, because the clustering algorithm proceeds along the gene array as ordered by Id, higher-numbered clusters will generally have higher variability. The ranks, and the clusters within them, may be considered as a matrix, and clusters may be uniquely identified by their position within the matrix — for example, the cluster of rank 2 with Id 47 is $C_{2,47}$.

With regards to computational complexity, the creation of clusters at each rank is of linear complexity, or $O(n)$. Because MAX_RANK is a constant, the entire clustering process is $O(\text{MAX_RANK } n) = O(n)$. If MAX_RANK is considered a variable, which in a sense it is because it depends on the structure of
the data, note that $\text{MAX\_RANK} \ll n$, and so clustering still approaches $O(n)$ performance. Calculation of cluster centers is also $O(n)$. Therefore, the most complex step in the algorithm remains the calculation of the correlation matrices as described above, and the SiMCAL 1 algorithm is $O(n^2)$ overall.

2.2.3 Linking

Linking proceeds in a similar fashion to clustering. Like neighbors, leaders and followers are ranked up to MAX\_RANK. A gene $x$ leads a gene $y$ at rank $r$ if $x$ is the rank $r$ leader of $y$; similarly, a gene $z$ follows $y$ if $z$ is the rank $r$ follower of $y$. This leads naturally to the concept of successions, the union of leaders and followers — in this example, $y$ succeeds $x$ and $z$ succeeds $y$. Also, $y$ is a successor
Now let $A$ and $B$ be two clusters of rank $R$, and $x \in A$ and $y \in B$. Then $A$ is a leader of $B$, and $B$ is a follower of $A$, if $y$ succeeds $x$ at rank $r \leq R$. In other words, for one cluster to lead another (and for the second cluster to be a follower of the first cluster; the two cases are identical) it is sufficient either for a gene in the first cluster to be a leader (at a rank equal to or less than that of the cluster) of a gene in the second cluster, or for a gene in the second cluster to be a follower of a gene in the first cluster.

Since clusters at higher ranks may be very large, containing numerous genes, some have many leaders and followers. This is desirable for the possible elucidation of biochemical pathways, since such pathways generally have many branch points. Representing the set of clusters as a directed graph allows navigation along many leader-follower paths through the graph, and comparison of those paths with biochemical pathways of interest.

Analysis of the linking yields some unexpected results. For instance, at rank 3 (the highest rank with more than one cluster) there is a single very large cluster (Cluster Rank 3, Id 0, or $C_{3,0}$) containing thousands of genes which showed little change in expression under experimental conditions. This is not surprising, as the majority of genes on the Affy mouse chip are presumably not involved in inflammatory response or other PSR-mediated activity.

What is surprising is that the majority of the other, much smaller clusters at rank 3 have $C_{3,0}$ as their sole leader and follower. This may imply that the rank 3 clusters are too large to be useful in analysis, and that the rank 1 and 2 clusters are more useful when trying to determine the biological meaning of the
clustering results. On the other hand, it may have some biological significance — perhaps implying that PSR-mediated reactions generally follow short pathways which involve few genes at a time.

2.3 Reporting

The reporting interface is Web-based and is designed to present a user-friendly means of analyzing the results. It provides an introductory page, general information about clusters and genes, the ability to search for and view specific clusters and genes, and a guide to interpretation (Fig. 2.4).

<table>
<thead>
<tr>
<th>Menu</th>
<th>Welcome to the SiMCAL1 gene expression platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>• &quot;Genes&quot; will show statistics about the genes</td>
</tr>
<tr>
<td></td>
<td>• &quot;Gene Search&quot; will provide an interface to search for genes</td>
</tr>
<tr>
<td>Genes</td>
<td>• &quot;Clusters&quot; will show statistics about the overall gene expression patterns in clusters</td>
</tr>
<tr>
<td>Clusters</td>
<td>• &quot;Cluster Search&quot; will provide an interface to search for clusters</td>
</tr>
<tr>
<td>Gene Search</td>
<td>• &quot;Logarithms&quot; will provide tables for converting expression values relative to baseline, and vice versa. You may select the baseline from the dropdown menu.</td>
</tr>
<tr>
<td>Cluster Search</td>
<td>A guide to usage:</td>
</tr>
<tr>
<td>Logarithms</td>
<td>• Buttons found as table headers may be used to sort or search results, save user annotations, or display other information.</td>
</tr>
<tr>
<td></td>
<td>• To open genes or clusters found in tables, click on the corresponding links.</td>
</tr>
</tbody>
</table>

![Figure 2.4: Part of the introductory screen.](image)

Expression levels of both genes and clusters are shown as measured data and interpolated values. The interpolation is via the cubic spline method [10, 238-248] which is a fairly robust and stable method. One problem with cubic splines is that they do not work well for unevenly spaced data; thus the values of $exp_0$, $exp_1$,
exp_2, and exp_3 are shown as evenly spaced over time, which may be deceptive if the user is not watching for this. On the other hand, the interpolated values, while they should certainly not assumed to be as accurate as the measured values, should be fairly representative of the actual activity of the gene over time.

Relationships between clusters, both hierarchical and leader-follower, may be navigated from any cluster (Fig. 2.5). The genes within the clusters are also available for view. Cluster, leader-follower, and neighbor relationships for genes may be similarly navigated. Any of these relationships can be shown on the same plot as the main entity under consideration, giving the user a quick means to visualize the relationships (Fig. 2.3).

**Figure 2.5:** Navigation through cluster relationships.
Both genes and clusters may be annotated by the user. In addition, the Affy annotation for genes is displayed when the gene is viewed. Unfortunately, the Affy annotations provided with the data are often not especially informative; [22] is recommended as a source for more complete annotations on genes of interest.

2.4 Biological Interpretation

A major challenge in biological analysis of the clustering results is in the quality of the data. The data used in this phase of the project have proven to have an unfortunately large number of measurements that were not deemed reliable enough to make it through preprocessing. As a practical matter, this means that many genes of interest were not included in the final data set.

The number of time-series measurements used here is small; although these particular time points were chosen deliberately based on the current understanding of the biological activity of PSR, a denser and more evenly spaced time series — perhaps one measurement at 30 minutes, followed by hourly measurements over the following 8, 9, or 10 hours — would be desirable for future study. It is worth noting that other experiments have had good results with small numbers of time series microarray measurements; an interesting example is [13], in which Hashimoto et al. compared expression levels of genes involved in spinal cord injury healing at 1, 3, and 7 days. However, more data can always provide more information, and it is to be hoped that higher-dimensional data on PSR activity will be available in the future.

Furthermore, the time course of 8 hours 30 minutes may have been too short to detect substantial associations between changes in gene activity relating to apoptosis and inflammatory response. Most studies in this area deal with time
courses of 24 hours or longer. Even with the existing limitations, however, some promising congruences with evidence from other sources did emerge.

2.4.1 Internal Analysis

\( C_{2,84} \) contains two genes which both hold significant interest and have Affy annotations which provide useful information about their function: \( G_{5285} \) and \( G_{5264} \). Both genes are highly upregulated under experimental conditions, and both show (as does \( C_{2,84} \)) a pattern of sharp upregulation early in the experiment followed by decline. (In Fig. 2.6, \( G_{5285} \) and \( G_{5264} \) are represented by the top and second-to-top blue lines, respectively. \( C_{2,84} \) is surrounded by other genes toward the middle of the plot.)

![Figure 2.6: \( C_{2,84} \) and its constituent genes.](image)

40
$G_{5285}$, which is known to be stimulated by PSR, is a gene for production of the protein prostaglandin (PG) G/H synthase, or PGH synthase, which is also known as COX2. This is a key enzyme in prostaglandin production. $G_{5264}$ is a gene for the PG receptor PGE2, and was not previously known to be stimulated by PSR. The fact that these genes show the same expression pattern may be an indicator that production of the PG receptor proceeds at the same time as the production of PG precursors, rather than following such production at a later time.

Another cluster of interest is $C_{2,29}$, which shows consistently high levels of expression throughout the experiment, with steadily increasing upregulation. Among its genes are $G_{5100}$, described by Affy as a “scavenger receptor,” and $G_{4974}$, described as a “house-keeping protein.” The presence of these genes in the same cluster, with consistently high expression on exposure to high concentrations of PSR, may indicate that they play a role in PSR mediation of immune response to apoptosis. Particularly interesting in this case is that $C_{2,29}$ is a follower of $C_{2,84}$.

$C_{2,29}$ and other followers of $C_{2,84}$ also contain several genes related to B cell production and activity. He et al. [14] found that “in the presence of resistant T cells, PGE2 increased B cell proliferation and differentiation.” Resistant T cells are defined here as cells which can increase immunoglobulin, or Ig, production; PSR antibody, of course, is an IgM antibody. Almost all B cells have IgM receptors. This is a good sign that the linking phase of the analysis correctly tracks the progression from PGE2 production to B cell proliferation.

$^4$Specifically, $G_{1004} \in C_{2,29}$, $G_{4544} \in C_{2,176}$, and $G_{3061} \in C_{2,229}$.
2.4.2 External Analysis

One useful way to measure the effectiveness of the clustering portion of the algorithm in biological terms is to measure results against other, more fully developed experiments. Stein et al. [34] studied gene regulation in the immune response to mammary involution, the process by which lactation ceases. This process naturally involves the apoptosis of large numbers of cells involved in milk production and secretion, and should not trigger a severe inflammatory response.

The same Affy chip, MG-U74Av2, as was used in this project, was subjected to experimental conditions related to mammary involution over 16 hours, and the results were analyzed by SOM (Self-Organizing Map) clustering using $\rho$ as the similarity measure. Genes, and the clusters they formed, which were not deemed of interest to the experiment were discarded from the final analysis. Eventually ten clusters of interest were found.

Although the experimental conditions were quite different from those used for this project, the biological processes under examination are similar; some correspondence of results should be apparent, and this is what was found. Genes which were clustered together in the Stein clusters were also clustered together by SiMCAL 1 at rank 2 in two of the ten Stein clusters, and at rank 3 in eight of the ten Stein clusters. The rate of agreement would probably have been still higher if many of the genes of interest had not been eliminated in SiMCAL 1 preprocessing.
3. SiMCAL 2

SiMCAL 2 has as its sole purpose the determination of differential gene expression under experimental conditions. Although a wide variety of statistical tools exist for determining differentiation in gene expression measurements (and, indeed, almost any set of data, of any kind) they all have limitations; see [26] and [33] for discussion of some of these. Therefore, it is to be hoped that SiMCAL 2 will make a real contribution to the body of work on this subject. Also, SiMCAL 2 offers a method of determining the degree of differentiation, a feature not found in most other such tests.

3.1 Data and Preprocessing

The SiMCAL 2 data are provided in a substantially different form from the data in SiMCAL 1; they have already been preprocessed to a significant degree, and furthermore, comprise four distinct data sets rather than one. Thus preprocessing consists mainly of getting the data into a usable form rather than performing calculations to normalize the measured values. Reliability information, although not as complete as that in the SiMCAL 1 data, is available for three of the four data sets. The data and the preprocessing methods used are thoroughly described in [33]; an overview is given here.

Also, unlike SiMCAL 1, SiMCAL 2 was developed from the beginning using simulated as well as real data. The simulated data, modeled as closely as possible on real data but with well-known properties, allowed thorough testing of the accuracy and performance of the algorithm. The nature of the simulated data
will be described in this section, and the testing performed will be referred to throughout the chapter.

All the “real” SiMCAL 2 data sets derive from the same source: measurements of gene expression activity in 12,225 genes ($n = 12225$) in 18 patients, 9 CF patients and 9 healthy control subjects, at a single time point ($m = 1$). Expression values are measured for perfect match (PM) oligonucleotides (“oligos”) which are expected to hybridize with the genes under examination and are used to measure actual gene expression, and for mismatch (MM) oligos which are not expected to hybridize with the genes under examination and are used to measure scanner noise and general chip activity not caused by experimental conditions. The data are then normalized in four different ways to produce distinct data sets, described in [33] as follows:

**MAS5** “MAS5 procedure uses the difference between PM and MM. The signal values of any hybridization are multiplied by a factor so that all chips to be analyzed have the same mean.”

**RMA** “RMA procedure ignores MM and removes global background. Normalization is done using quantile normalization, which is a method in which probe intensities are adjusted to produce identical distributions. RMA procedure has been obtained from http://www.bioconductor.org.”

**Dchip** “Dchip procedure uses a baseline array to normalize arrays by selecting invariant sets of genes(or probes) then using them to fit a non-linear relationship between the ‘treatment’ and ‘baseline’ arrays. The non-linear relationship is used to carry out the normalization. This procedure can be
based on PM intensity only or PM/MM intensity differences. Dchip method is available at http://www.dchip.org.”

Note that the MAS5 and RMA procedures produce only one data set apiece, here referred to simply as MAS5 and RMA, while the Dchip procedure produces two: DchipPM and DchipPM_MM.

3.1.1 Measurement Reliability and Combination

None of the four data sets have p-values provided; however, all but RMA do have Affy evaluations of “present”, “absent”, or “marginal” for each gene in each patient. Therefore, a simple standard is used for determining gene expression in MAS5, DchipPM, and DchipPM_MM: to be included in the final analysis, a gene must have at least one expression measurements in the healthy subjects and one in the CF patients which is considered to be “present”, and \( \text{exp}_{\text{normal}} \) is set equal to the mean of the “present” healthy subject expressions for that gene, while \( \text{exp}_{\text{abnormal}} \) is set equal to the mean of the “present” CF patient expressions for that gene. The procedure for RMA is even simpler — normal and abnormal expressions for each gene are determined by finding the mean of all healthy and CF expression values, respectively, for that gene.

This leads naturally to the definition of some data structures:

- The Entity data structure has the attributes \( \text{Expression} \), a floating-point value which is calculated as described above, and \( \text{IsNormal} \), a Boolean value describing whether a given instance of Expression represents expression in healthy subjects (\( \text{IsNormal} = \text{True} \)) or CF patients (\( \text{IsNormal} = \text{False} \)). Entity also has the attribute \( \text{Rank} \), an integer value whose use will be de-
scribed later. At this point, Entity.$\text{Rank} = 0$ for all Entities created so far.

- The Gene data structure has various attributes relating to annotation, and of more interest at this point, the $\text{NormalEntity}$ and $\text{AbnormalEntity}$ attributes, whose names should be self-explanatory.

Genes and entities (and later, clusters) are assigned Id numbers and are referenced as described in Ch. 2.1.3.

### 3.1.2 Algorithm Outline

At this point, it is appropriate to look at the overall structure of the SiMCAL 2 algorithm, presented here in pseudocode:

1. \( G = \) all genes in data set
2. \( E = \) all entities (normal and abnormal) attached to genes in \( G \)
3. while \( E \) is not empty:
   4. find nearest neighbor for each entity in \( E \)
   5. cluster based on nearest neighbors
   6. find cluster intersections
   7. \( G = \) differentially expressed genes based on clusters
   8. \( E = \) entities attached to genes in \( G \)
   9. if \( E = \) previous iteration’s \( E \):
      10. end while loop

Lines 1-4 have been covered in the preceding section; lines 5-10 are explained in the following sections.
3.1.3 Simulated Data

Simulated data can consist of any number \( (n) \) of normal and abnormal expression values, which then generate data structures as described above. A wide range of values was used during development and testing, \( n \in [100, 10000] \).

One advantage of the non-parametric nature of SiMCAL analysis is that it does not incorporate the assumption, common to many statistical tests of differentiation, that the data are normally distributed\(^2\). The simulated data can, of course, have any distribution desired, as long as the mean and variance are reasonably close to those of real data. Most testing with simulated data was performed using a mixture model in which noise has a normal distribution while up- and down-regulation under experimental conditions have a uniform distribution. Other distributions, including exponential and beta, for both noise and regulation were used in testing at various points, and all produced good results.

3.2 Clustering and Linking

Clustering in SiMCAL 2 is based on the same general principle as in SiMCAL 1, but proceeds rather differently. The most important difference is that a number of genes are eliminated at each iteration: since the purpose of SiMCAL 2 analysis is to determine which genes are differentially expressed, those genes found not to be differentially expressed at each rank are eliminated from consideration at higher ranks, and are not included in the clustering. (Recall that in SiMCAL 1, all genes in the data set are included in the clustering at all ranks.) Also, clustering is generally faster and simpler, for reasons which are described below.

3.2.1 Distance Measures and Matrices

Because \( m = 1 \), a correlation measure between two expressions such as the \( \rho \) used in SiMCAL 1 is clearly meaningless. Instead, given two entities \( E_1 \) and
$E_2$, with expressions $e_1$ and $e_2$, the distance between the entities is simply the absolute value of the difference of their expressions:

$$\Delta(E_1, E_2) = |e_1 - e_2|. \tag{3.1}$$

It can be seen that this is actually a special case of the city block distance $\Delta_1$ (Eqn. 1.15) in which the expression “vectors” consist of a single value.

Now let $G_r$ be the vector of genes under consideration at rank $r$, and let $n_r = |G_r|$, thus $G_r = (G_{r,1}, G_{r,2}, \ldots, G_{r,n_r})$. The expressions of all entities attached to these genes are considered as the vector $\text{exp}_r$ of length $2n_r$, because each gene has two expressions, one normal and one abnormal:

$$\text{exp}_{r,i} = G_{r,i} . \text{NormalEntity.Expression};$$
$$\text{exp}_{r,i+1} = G_{r,i} . \text{AbnormalEntity.Expression}; \tag{3.2}$$
$$\text{exp}_r = (\text{exp}_{r,1}, \text{exp}_{r,2}, \ldots, \text{exp}_{r,2n_r}).$$

Then the distance vector for any entity $E_{r,i}$, $i \in [1, 2n_r]$, expressed as a vector-scalar function which follows from Eqns. 1.3 and 3.1, is:

$$\text{dists}_{r,i}(E_{r,i}) = \Delta(\text{exp}_r, \text{exp}_{r,i}). \tag{3.3}$$

Next, the $i$th element of $\text{dists}_{r,i}$ is set equal to $\max(\text{dists}_{r,i})$ in order to avoid assigning $E_{r,i}$ as its own nearest neighbor, and then the nearest neighbor of $E_{r,i}$ is:

$$\text{nearest}(E_{r,i}) = \arg\min(\text{dists}_{r,i}). \tag{3.4}$$
The neighbor-finding operation, because it must find distances between each of \(2n_r\) entities and each other entity in the set under consideration, is \(O(n_r^2)\). Also, because \(n_r\) is an approximately linear function of \(n\), the number of genes in the original data set, neighbor-finding is ultimately \(O(n^2)\). However, there are two important and non-obvious features to consider in this analysis:

1. Calculating \(\Delta\) is lighter-weight than calculating \(\rho\), and so generally SiMCAL 2 neighbor-finding proceeds a good bit faster than the equivalent operation in SiMCAL 1.

2. For any \(r > 1\), it is likely that \(n_r \ll n\), so operations at higher ranks are particularly fast.

Details of performance are given later in this chapter.

Note that leaders and followers are not found here as they are in SiMCAL 1. Again, this is simply because \(m = 1\) — without measured changes in the behavior of the genes over time, there is no meaningful way to assign succession relationships between them.

### 3.2.2 Ranks and Cluster Formation

SiMCAL 2 clustering is different in a number of ways from that in SiMCAL 1. It is simpler and lighter-weight, because in the previous step, only a single nearest neighbor was discovered for each entity. Also, the specific programming logic used is somewhat different, because the set of entities under consideration changes at each iteration (i.e., at each rank). Perhaps the most significant difference is that \textit{entities} are clustered, not \textit{genes}; a gene is a member of a cluster if either or both
of its attached entities are a member of that cluster. However, the underlying concepts are the same.

The SiMCAL 2 clustering implementation depends heavily in the Python “set” data type, which does just what the name implies: it is an unordered collection of data, to which may be applied any of the usual set-theoretical operations such as intersection, union, symmetric difference, etc. Operations on sets are generally much faster than operations on arrays, vectors, etc. in Python or in other programming languages, because set members are stored in hash tables which (theoretically) offer nearly $O(1)$ lookup times. Testing shows that given two sets $S$ and $T$, operations involving either $S$ or $T$ by itself are effectively $O(1)$, and operations involving both sets (e.g., finding their intersection) are roughly $O(\min(|S|,|T|))$.

The first step is to put each the nearest neighbor of each entity into a single-member set. Then, the following Python code from the clustering control file shows the core of the clustering procedure, at any rank:

```python
def main:
    for Id in Clusters:
        if Touched[Id] is None:
            Candidates = Clusters[Id]
            NewCluster = makeset(Id) # make sure the Entity under
            # examination is in own cluster
            while len(Candidates) > 0:
                next = Candidates.pop()
                if next not in NewCluster:
```

50
NewCluster.add(next)

if Touched[next] is None:
    Candidates.update(Clusters[next])
    Clusters[next].clear()
else:
    Candidates.add(Touched[next])
    Touched[next] = Id
    Clusters[Id].update(NewCluster)

At the beginning of execution, Touched is simply an empty array of length $2n_r$ containing None (the Python equivalent of NULL in C) values. The make-set() function takes a single data object as an argument and returns a single-member set containing that object as its only element; $S = \text{make.set}(s) \equiv S = \{s\}$. The pop() method removes a randomly selected element from a set and returns that element, while the add() method adds the element which is passed as an argument. The update() method performs a union; $S.\text{update}(T) \equiv S = S \cup T$. Note that because these are true sets, duplicate values are eliminated — e.g., if $S = \text{set}([1, 2, 3])$ and $T = \text{set}([2, 3, 4])$, then $S.\text{update}(T)$ will result in $S = \{1, 2, 3, 4\}$, and add() produces similarly correct results.

The end result is that Clusters contains a large number of empty sets and a few sets containing large numbers of entities, which have been clustered together with their nearest neighbors. Only these non-empty sets are saved to the database as clusters. The Cluster data structure, in addition to Id and Rank, also has the attribute AttachedEntity — this entity represents the cluster center, and, not surprisingly, has its Expression attribute set equal to the mean of the
Expressions of the entities in the cluster.

3.2.3 Linking

Linking in SiMCAL 2 is based on the concept of cluster intersections. As seen above, a cluster may be considered as a set of entities. It may also, however, be considered as a set of genes. For any gene $G$ and cluster $C$:

$$G.\text{NormalEntity} \in C \lor G.\text{AbnormalEntity} \in C \Rightarrow G \in C. \quad (3.5)$$

Then, considering any two clusters of equal rank $C$ and $D$ as sets of genes, $C$ and $D$ are intersecting clusters if $C \cap D \neq \emptyset$. The congruence of $C$ and $D$ is as given in Eqn. 1.16. (This comparison can only be done using clusters as sets of genes; clusters as sets of entities are always disjoint.) Therefore, although there is no temporal linking, relationships between clusters at the same rank are established. SiMCAL 2 users can navigate between intersecting clusters in the same way as SiMCAL 1 users can navigate cluster successions.

Parent and child relationships are also formed between clusters at different ranks, just as in SiMCAL 1; however, the definition is slightly different. Consider two clusters $E$ and $F$, where $E.\text{Rank} = r$ and $F.\text{Rank} = r + 1$, as sets of entities. In SiMCAL 1, it was the case that $F$ was the parent of $E$, and $E$ was a child of $F$, if $E \subset F$. However, since in SiMCAL 2, some genes (and thus their entities) are eliminated at each iteration, there may be elements of $E$ which are not present in $F$ or in any other cluster at rank $r + 1$. Therefore, here $F$ is the parent of $E$, and $E$ is a child of $F$, if $E \cap F \neq \emptyset$. 

52
3.3 Differentiation

Differentiation — that is, the determination of which genes are differentially expressed under experimental, as opposed to control, conditions — is the purpose of SiMCAL 2 analysis. The condition used for differentiation has evolved somewhat during the history of this phase of the project, and provides the best way to understand the current condition.

There exist two major measures of accuracy in any differentiation test: sensitivity, the degree to which the test manages to identify those data elements which are meaningfully “different” from the control set; and specificity, the degree to which data elements identified as different in fact are. Colloquially, low sensitivity means a high rate of “false negatives” while low specificity means a high rate of “false positives.” Obviously, both sensitivity and specificity should be as high as possible; depending on the precise application, one or the other may be of greater importance, but neither should be sacrificed. In the current application, specificity may be slightly more important simply because of the size of the data set — it is more useful to identify a few genes which are almost certainly differentially expressed, for further study, than to identify a great many genes which may be differentially expressed but whose number still leaves the user drowning in data.

Fortunately, the use of simulated data allows precise testing of both accuracy measures for this algorithm. (This is a major improvement over SiMCAL 1.) The simulation code generates a set of genes which includes “abnormal” genes — those which are differentially expressed — which here is denoted $G_A$. Then the algorithm itself, at each rank $r \in [1, r_{\text{max}}]$, finds a set of differentially expressed genes: $G_{D,r}$. (Recall that $r_{\text{max}}$ is not set by the user a priori, as in SiMCAL 1;
instead, it is simply, by definition, the maximum rank at which any differentially expressed genes are found.) This provides a simple measure of both sensitivity and specificity for any value of $r$:

\[
\begin{align*}
\text{sens}_r &= \frac{|G_A \cap G_{D,r}|}{|G_A|}; \\
\text{spec}_r &= \frac{|G_A \cap G_{D,r}|}{|G_{D,r}|}.
\end{align*}
\] (3.6)

The ideal result for SiMCAL 2 processing would be to have perfect sensitivity and good specificity at lower ranks, with acceptable sensitivity and perfect specificity at higher ranks; the idea is that as $r$ increases, only the genes with greater differentiation will be identified as differentially expressed. Therefore, identification at higher ranks should serve as a guide to the degree of differential expression — or, in the context of the current data set, the severity of a gene’s up- or down-regulation under disease conditions.

The original condition for differentiation was very simple. At any rank $r$, any gene $G$ under consideration (which is any $G \in G_{D,r-1}$) may be said to belong to two clusters, $N(G)$ and $A(G)$, where $G.\text{NormalEntity} \in N(G)$ and $G.\text{AbnormalEntity} \in A(G)$. Thus the condition was:

\[
N(G) \neq A(G) \Rightarrow G \in G_{D,r}.
\]

Unfortunately, this condition produces very high sensitivity ($\text{sens}_r \approx 1$ at all ranks) but very low sensitivity, and is therefore insufficient. The clear answer is to provide some sort of filter to remove the large number of false positives found. The next step, therefore, is to introduce the idea of a distance matrix between the all clusters at rank $r$, which is denoted $C_r = (C_{r,1}, C_{r,2}, \ldots, C_{r,|C_r|})$. Then the
distance matrix for all clusters at rank $r$ is:

$$
\text{Cdist}_{r,i,j} = \Delta(C_{r,i}, \text{AttachedEntity}, C_{r,j}, \text{AttachedEntity}).
$$

The next step is to scale this matrix so that all values are within the range $[0, 1]$, thus producing the *scaled distance matrix*:

$$
\text{SCdist}_{r,i,j} = \frac{\text{Cdist}_{r,i,j}}{\max(\text{Cdist})}
$$

For notational convenience, given clusters $C_{r,i}$ and $C_{r,j}$, let $\Delta_S(C_{r,i}, C_{r,j}) = \text{SCdist}_{r,i,j}$. Then the condition for differentiation became:

$$
\Delta_S(N(G), A(G)) > \text{Con}(N(G), A(G)) \Rightarrow G \in G_{D,r}.
$$

Note that the original condition is subsumed by this condition:

$$
N(G) = A(G) \Rightarrow \Delta_S(N(G), A(G)) = 0 \land \text{Con}(N(G), A(G)) = 1
$$

... and therefore ...

$$
\Delta_S(N(G), A(G)) > \text{Con}(N(G), A(G)) \Rightarrow N(G) \neq A(G).
$$

However, it is clearly *not* the case that:

$$
N(G) \neq A(G) \Rightarrow \Delta_S(N(G), A(G)) > \text{Con}(N(G), A(G)).
$$

Thus this condition is much more restrictive, as desired.

The justification for the new condition is that if two clusters have a high distance between them, and low congruence, then any genes shared between them are likely to be genuinely differentially expressed; on the other hand, clusters
which are close together and have a high congruence may well be in some sense the “same” cluster. That is, it is quite likely that some entity in one cluster is a very near neighbor of some entity in the other cluster (though not, of course, the nearest neighbor; in that case the clusters would not be separate at all) and so genes shared between the clusters are not actually differentially expressed. As a practical matter, using this condition dramatically improves specificity \((spec_r \approx 1\) at all ranks) while leading to noticeably diminished but perhaps still acceptable sensitivity \((sens_r > 0.5\) at most ranks.)

It should still be possible to do better. The fact that specificity is now very high while sensitivity is lower implies that perhaps the new condition is new restrictive. The solution is to apply a scaling factor, \(\kappa\), to the congruence before performing the test. After trying a variety of values, the best results were obtained with the following condition:

\[
\kappa = \frac{1}{3}; \quad \Delta_S(N(G), A(G)) > \kappa \times \text{Con}(N(G), A(G)) \Rightarrow G \in G_{D,r}.
\]

Results here are nearly ideal for small \((n < 1000)\) data sets. At lower ranks, \(sens_r \approx 1\) and \(spec_r > 0.75\). As \(r \to r_{\text{max}}, spec_r \to 1\). Sensitivity decreases (although rarely goes below 0.5) at higher ranks, but this is to be expected and is in fact desirable, because only the more differentially expressed genes are identified as such at these ranks. Using \(\log\)-scaled values for expression changes between normal and abnormal entities attached to the same gene, such that a change of 1 indicates a twofold change in expression, 2 indicates a fourfold change, etc., minimum values for expression change in genes identified as differentially expressed climb from \(\leq 1\)
at \( r = 1 \) to \( \geq 5 \) at \( r = r_{\text{max}} \), while mean values climb from \( \leq 5 \) at \( r = 1 \) to \( \geq 8 \) at \( r = r_{\text{max}} \). As the last value is about equal to the greatest observed change in both real and simulated data, this is an excellent accuracy figure.

However, with large data sets (especially \( n > 3000 \)) a problem emerges. A desirable value for \( r_{\text{max}} \) is somewhere in the neighborhood of 3 to 5, and certainly values higher than 7 or so are not desirable, but the value of \( \kappa \) used above can lead to 10 or more ranks being formed. Recall that the second SiMCAL condition states that levels (ranks) should be few relative to hierarchical clustering, so that the user does not have to decide on an arbitrary cut level to interpret the analysis. Clearly 10 or even 20 ranks is preferable to the hundreds of levels which standard hierarchical clustering can produce, but it is still too large a number to allow for easy interpretation.

The solution is to make \( \kappa \) a variable, not a constant, and dependent on rank. The value of \( \frac{1}{3} \) used above is good for the first rank, but is inadequate for the higher ranks which SiMCAL 2, in this form, will try to create. The condition must become more restrictive as ranks increase. Therefore, the final form of the condition which determines whether a gene is differentially expressed is:

\[
\kappa = \frac{2^{r-1}}{3}; \\
\Delta_S(N(G), A(G)) > \kappa \times \text{Con}(N(G), A(G)) \Rightarrow G \in G_{D,r}. \quad (3.9)
\]

Once \( G_{D,r} \) is determined, the algorithm iterates again at \( r + 1 \) with the entities attached to the genes in \( G_{D,r} \). The entire algorithm halts when either \( |G_{D,r}| = 0 \) or \( G_{D,r} = G_{D,r-1} \).
### Table 3.1: SiMCAL 2 accuracy figures.

No particular justification is offered for the formula used to determine $\kappa$ in Eqn. 3.9, except that it works. Sensitivity at $r = 1$ is excellent, and specificity very good; at higher ranks, specificity rapidly reaches 1, and while sensitivity drops, the genes which are identified as differentially expressed are uniformly those with greater differences between normal and abnormal expressions. Just as importantly, the number of ranks formed is constrained — even for the largest real data set, RMA with $n = 12625$, operations terminated at $r_{max} = 5$. Accuracies for a typical run with simulated data, $n = 3000$, are shown in Table 3.1.

#### 3.4 Reporting

The reporting interface in SiMCAL 2 is substantially similar to that in SiMCAL 1, although with a few refinements to make it easier to use, such as the simplified search interface show in Fig. 3.1.

The primary difference is in visualization of genes and clusters. Because $m = 1$, time-series plots would of course not be meaningful. Instead, the standard adopted for display is:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Min. Differential</th>
<th>Mean Differential</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.902</td>
<td>0.993</td>
<td>0.285647</td>
<td>5.37723</td>
</tr>
<tr>
<td>2</td>
<td>0.81</td>
<td>1</td>
<td>0.791236</td>
<td>5.85364</td>
</tr>
<tr>
<td>3</td>
<td>0.613</td>
<td>1</td>
<td>0.791236</td>
<td>6.62608</td>
</tr>
<tr>
<td>4</td>
<td>0.325</td>
<td>1</td>
<td>2.43204</td>
<td>7.56764</td>
</tr>
<tr>
<td>5</td>
<td>0.072</td>
<td>1</td>
<td>5.61399</td>
<td>8.42569</td>
</tr>
</tbody>
</table>
1. Genes and clusters are both displayed as points on a 2-dimensional graph.

2. For genes, $\text{NormalEntity.Expression}$ is displayed on the x-axis, while $\text{AbnormalEntity.Expression}$ is displayed on the y-axis. Fig 3.2 shows expressions for all genes in the DchipPM_MM data set, distinguishing between genes found to be differentially expressed and those not found to be differentially expressed.

3. For clusters, since they have only one entity (and thus, one expression) which represents the cluster center, $\text{AttachedEntity.Expression}$ is displayed on both the x- and y-axes. Fig. 3.3 shows all clusters in DchipPM_MM at all ranks, against a background of genes in the data set.

3.5 Performance

SiMCAL 2 is very fast compared to SiMCAL 1, in large part due to the relative simplicity of its operations. It is worth analyzing each step and then seeing how the algorithm performs in practice.

It is clear that neighbor-finding operations are inherently $O(n^2)$, but in practice they tend to follow a polynomial of lower order, about $O(n^{1.7})$. This is probably due to the efficiency of the NumPy routines used for vector calculation,
which are used to obtain the results in Eqns. 3.3 and 3.4. It seems likely that with very large data sets, these operations would eventually become truly $O(n^2)$, but this would be with much larger data sets than any actually used here.

Clustering is roughly $O(n)$. This is because early in the clustering, clusters grow rapidly, effectively removing entities from future clustering operations once they are already in a cluster. There is a pathological case in which clustering is $O(n^2)$, but this simply cannot occur with real (or realistic simulated) data — it happens only when every entity has the same expression value! In this case, each entity reports $E_{r,1}$ as its own nearest neighbor, and thus entities are removed from
consideration during clustering operations only one at a time.

Since the number of clusters $|C_r|$ is roughly a linear function of the number of genes, and since differentiation at each rank requires the construction and analysis of $|C_r| \times |C_r|$ matrices, it might also be expected that differentiation is $O(n^2)$. However, not only does this step take heavy advantage of NumPy’s vector routines and Python’s set operations, both highly optimized, but it also is written to do as much as possible of the work by iterating through genes instead of comparing clusters directly. Thus, like neighbor-finding, in practice it tends to follow a lower-order polynomial, in this case about $O(n^{1.2})$. 

Figure 3.3: Clusters and genes in DchipPM.MM.
Table 3.2 shows operation times for $n \in [200, 3000]$. Total operation times may not be precisely equal to the sum of neighbor-finding, clustering, and differentiation times, especially at lower ranks, because all times are rounded to the nearest integer value.

Fig. 3.4 shows the execution times for the various components of the algorithm plotted against their approximate orders, while Fig. 3.5 shows total execution time. Unsurprisingly, total execution time also has a polynomial order $< O(n^2)$, about $O(n^{1.3})$. Again, of course, it is reasonable to believe that this figure would increase for very large data sets, but there are no data sets this large currently under consideration.

![Figure 3.4: Execution times for SiMCAL 2 components.](image)

Figure 3.4: Execution times for SiMCAL 2 components.
<table>
<thead>
<tr>
<th>n</th>
<th>Neighbors</th>
<th>Clusters</th>
<th>Differentials</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>400</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>600</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>800</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>1000</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>23</td>
</tr>
<tr>
<td>1200</td>
<td>8</td>
<td>7</td>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>1400</td>
<td>11</td>
<td>9</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>1600</td>
<td>13</td>
<td>10</td>
<td>17</td>
<td>41</td>
</tr>
<tr>
<td>1800</td>
<td>15</td>
<td>11</td>
<td>19</td>
<td>46</td>
</tr>
<tr>
<td>2000</td>
<td>18</td>
<td>12</td>
<td>22</td>
<td>52</td>
</tr>
<tr>
<td>2200</td>
<td>21</td>
<td>13</td>
<td>25</td>
<td>60</td>
</tr>
<tr>
<td>2400</td>
<td>25</td>
<td>15</td>
<td>28</td>
<td>68</td>
</tr>
<tr>
<td>2600</td>
<td>28</td>
<td>17</td>
<td>32</td>
<td>78</td>
</tr>
<tr>
<td>2800</td>
<td>33</td>
<td>18</td>
<td>35</td>
<td>86</td>
</tr>
<tr>
<td>3000</td>
<td>35</td>
<td>19</td>
<td>37</td>
<td>92</td>
</tr>
</tbody>
</table>

**Table 3.2:** SiMCAL 2 run times, in seconds.
3.6 Biological Interpretation

It was found in [33] that “the choice of normalization seems to greatly affect the differentially expressed genes across the conditions.” In other words, for each of the four data sets, greatly varying numbers of differentially expressed genes were discovered. Similar results were found here. Table 3.3 shows what varying results were obtained.

A further refinement is to see how many genes the datasets had in common initially (that is, after preprocessing) as shown in Table 3.4, and how many genes they had in common which were identified as differentially expressed, as shown in

Figure 3.5: Execution times for SiMCAL 2 overall.
### Table 3.3: Total genes under consideration, and differential genes discovered.

<table>
<thead>
<tr>
<th></th>
<th>MAS5</th>
<th>DchipPM</th>
<th>DchipPM_MM</th>
<th>RMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genes</td>
<td>6219</td>
<td>7658</td>
<td>7658</td>
<td>12625</td>
</tr>
<tr>
<td>Rank 1 Differentials</td>
<td>168</td>
<td>965</td>
<td>1681</td>
<td>450</td>
</tr>
<tr>
<td>Rank 2 Differentials</td>
<td>105</td>
<td>248</td>
<td>636</td>
<td>102</td>
</tr>
<tr>
<td>Rank 3 Differentials</td>
<td>53</td>
<td>66</td>
<td>202</td>
<td>28</td>
</tr>
<tr>
<td>Rank 4 Differentials</td>
<td>19</td>
<td>11</td>
<td>45</td>
<td>3</td>
</tr>
<tr>
<td>Rank 5 Differentials</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.5.

Ultimately, only three genes were identified as differentially expressed in all datasets. The Affy designations for these genes are “38095_i.at”, “32794_g.at”, and “AFFX-HSAC07/X00351_3.at”. They are described at [22]:

**38095_i.at** A major histocompatibility complex (MHC) gene responsible for pathogen and antigen detection. CF often involves an exaggerated (although ultimately ineffective) immune response. Particularly interesting is that this gene is involved in cross-membrane transport; recall that CF is largely a disorder of this process.

**32794_g.at** A T-cell (immune cell) receptor gene, also membrane-associated, located on chromosome 7, where the mutations believed to be responsible for CF occur.

**AFFX-HSAC07/X00351_3.at** Involved primarily in cellular motor activity.
### Table 3.4: Genes shared by datasets after preprocessing.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAS5</th>
<th>DchipPM</th>
<th>DchipPM_MM</th>
<th>RMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAS5</td>
<td>-</td>
<td>6076</td>
<td>6076</td>
<td>6186</td>
</tr>
<tr>
<td>DchipPM</td>
<td>6076</td>
<td>-</td>
<td>7658</td>
<td>7619</td>
</tr>
<tr>
<td>DchipPM_MM</td>
<td>6076</td>
<td>7658</td>
<td>-</td>
<td>7619</td>
</tr>
<tr>
<td>RMA</td>
<td>6186</td>
<td>7619</td>
<td>7619</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 3.5: Differentially expressed genes shared by datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAS5</th>
<th>DchipPM</th>
<th>DchipPM_MM</th>
<th>RMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAS5</td>
<td>-</td>
<td>10</td>
<td>21</td>
<td>28</td>
</tr>
<tr>
<td>DchipPM</td>
<td>10</td>
<td>-</td>
<td>418</td>
<td>31</td>
</tr>
<tr>
<td>DchipPM_MM</td>
<td>21</td>
<td>418</td>
<td>-</td>
<td>65</td>
</tr>
<tr>
<td>RMA</td>
<td>28</td>
<td>31</td>
<td>65</td>
<td>-</td>
</tr>
</tbody>
</table>
It is associated with a cytoskeletal acetyltransferase complex; CF patients metabolize acetyltransferase-associated drugs unusually fast. This gene is also located on chromosome 7.

Therefore, although no definitive conclusions can be drawn from these results, it is reasonable that all three of these genes should be strongly differentially expressed in CF patients. More generally, genes which are found to be differentially expressed at high ranks in any of the data sets (particularly in \textbf{MAS5}, considered in [33] to use the best of the normalization methods) should be considered for more intensive future research.
4. SiMCAL 3

SiMCAL 3 combines the major features of SiMCAL 1 and 2 into a single application. Like SiMCAL 1, it works with time-series data \((m > 1)\) by clustering genes over time and finds succession relationships between genes and clusters. Like SiMCAL 2, it determines which genes in the data set are differentially expressed and offers the feature of determining the degree of differentiation. It offers advantages over both its predecessors: it is designed to work with any value of \(m\) rather than a specific value for a specific data set \((m = 3 \text{ for SiMCAL 1, } m = 1 \text{ for SiMCAL 2})\) and it is to be hoped that the elimination of genes which are not differentially expressed will lead to more meaningful clustering and linking. SiMCAL 3 is capable of reproducing the features of both of its predecessors, but offers analytical power beyond either of them.

4.1 Data and Preprocessing

As with SiMCAL 2, SiMCAL 3 was developed using simulated data, and then used to analyze real data. The real data come from the SiMCAL 1 data set, but are preprocessed in a slightly different way. After eliminating unreliable measurements, there exists the same set of measurements given in Eqn. 2.1. However, in SiMCAL 3, normalization proceeds as follows, generating both normal and abnormal expression vectors for each gene in each replicant:

\[
\begin{align*}
\exp_t^{normal} &= \lg \frac{\text{isol}}{\text{control}}, \ t \in [1, m]; \\
\exp_t^{abnormal} &= \lg \frac{\text{psr}}{\text{control}}, \ t \in [1, m].
\end{align*}
\]
These vectors are then averaged across replicants to produce expression vectors for the normal and abnormal entities attached to each gene. Reliability measures are not calculated or used as weighting factors, in keeping with the practice used in SiMCAL 2 — it is reasonable to expect that any measurement with $p \leq 0.05$, which is marked “present” by the Affy equipment, is reliable enough to be used as in in future calculations.

The rather artificial “$\text{exp}_0$” data point is not used in most of SiMCAL 3, although it does resurface in succession-finding, as explained in following sections.

Data structures are essentially identical to those used in SiMCAL 2, except that Entity now has the attribute $\text{Expressions} = (\text{exp}_1, \text{exp}_2, \ldots, \text{exp}_m)$. This is a vector (as calculated in Eqn. 4.1 in the case of entities attached to genes) of floating-point values instead of the scalar $\text{Expression}$ used in SiMCAL 2.

4.1.1 Algorithm Outline

Unsurprisingly, the large-scale logic of SiMCAL 3 is basically a combination of that in its predecessors:

1. $G =$ all genes in data set
2. $E =$ all entities (normal and abnormal) attached to genes in $G$
3. while $E$ is not empty:
4. find nearest neighbor and successions for each entity in $E$
5. cluster based on nearest neighbors
6. find cluster intersections and successions based on succession relationships of member entities
7. $G =$ differentially expressed genes based on clusters
8. $E =$ entities attached to genes in $G$
if $E =$ previous iteration’s $E$:

end while loop

4.1.2 Simulated Data

Simulated data were created here by drawing randomly selected normal expression values from the real data and then applying a variety of distributions for up- and down-regulation. A uniform distribution seems to be closest to the differential expressions in the real data, but others were tested with good results, including normal, beta, and exponential.

4.2 Clustering and Linking

Clustering is almost identical to that in SiMCAL 2. However, SiMCAL 3 of course has a succession-finding component which is wholly lacking in SiMCAL 2, and the fact that the algorithm deals with multidimensional data requires the use of more sophisticated distance measures. One of the driving forces in development of SiMCAL 3 has been to incorporate the lessons learned in the development of SiMCAL 2 which make that algorithm superior to SiMCAL 1, while retaining the features of both.

One lesson in the development of SiMCAL 3, while testing against a variety of simulated data, was that despite the arguments for the use of $\rho$ as a similarity measure for clustering which are detailed in Ch. 2.2.1, $\rho$ is a poor choice for determining differential expression. If the goal is simply to cluster genes together based on behavior, $\rho$ is still an excellent standard; but when vagaries of the clustering process may lead to genes being eliminated from the data set, as is the case in both SiMCAL 2 and SiMCAL 3, accidental correlation in noisy data can
lead to wildly inaccurate results. Specifically, when using \( \rho \) in early development of SiMCAL 3, both sensitivity and specificity were observed in the range of 0.3 to 0.5 at all ranks, which is clearly unacceptable.

A variety of other distance and similarity measures were tried.\(^5\) Other correlative measures, including covariance and Spearman correlation, failed to produce significantly better results. A return to Minkowski distances (Ch. 1.3) was tried next: first \( \Delta_2 \), the Euclidean, which produced good but not spectacular results (both sensitivity and specificity in the 0.5-0.8 range) and finally, the simplest possible Minkowski distance for multidimensional data, \( \Delta_1 \), the city block distance. Perhaps unsurprisingly, since a special case of this measure was used in SiMCAL 2 with great success, it was \( \Delta_1 \) that produced the best results. Genes with generally similar behavior over time were clustered together as in SiMCAL 1, and determination of differential expression was as good as in SiMCAL 2.

However, \( \Delta_1 \), and Minkowski measures generally, proved to be very poor at succession-finding, and so after further experimentation, \( \rho \) was retained for this purpose. This leads to the necessity of defining two separate measures to be used in SiMCAL 3:

- The standard distance measure, used for neighbor-finding, which is defined in the current form of the algorithm as \( \Delta_1 \); and

- The standard similarity measure, used for succession-finding, which is defined in the current form of the algorithm as \( \rho \).

\(^5\) Naturally, very slight alterations to the code are necessary for neighbor-finding operations depending on whether a distance measure or a similarity measure is used, but the underlying logic is the same.
For maximum flexibility, so that users can focus on specific facets of gene behavior as desired, the application is written to allow using different standard distance and/or similarity measures with very minor code changes. However, using the definitions of these measures given above is strongly recommended for the best results with a wide variety of data sets.

Neighbor-finding proceeds much as it does in SiMCAL 2 (Eqns. 3.2 - 3.4) with the sole exception that $\exp_r$ is now a matrix rather than a vector, and $\exp_{r,i}$ is now a vector rather than a scalar. This implies a slight change in notation: let $\exp_{r,1}$ represent $E_{r,i}$. Expressions, and the matrix of expressions for all entities under consideration is:

$$\text{Exp}_r = (\exp_{r,1}, \exp_{r,2}, \ldots, \exp_{r,2n_r}).$$  \hspace{1cm} (4.2)

Then the distance vector for any entity $E_{r,i}, i \in [1, 2n_r]$ is:

$$\text{dists}_{r,i}(E_{r,i}) = \Delta_1(\text{Exp}_r, \exp_{r,i}).$$  \hspace{1cm} (4.3)

The nearest neighbor is then found as in Eqn. 3.4.

Succession-finding is very similar to that in SiMCAL 1, with some exceptions. For any $\exp_{r,i} = (\exp_{r,i,1}, \exp_{r,i,2}, \ldots, \exp_{r,i,m})$, let:

$$\exp_{r,i}^{\text{early}} = (0, \exp_{r,i,1}, \exp_{r,i,2}, \ldots, \exp_{r,i,m-1}).$$  \hspace{1cm} (4.4)

It follows that:

$$\text{Exp}_r^{\text{early}} = (\exp_{r,1}^{\text{early}}, \exp_{r,2}^{\text{early}}, \ldots, \exp_{r,2n_r}^{\text{early}}).$$  \hspace{1cm} (4.5)
This is the only case in which \( \text{"exp\_0"} \) is introduced; it plays no part in neighbor-finding or clustering, and is not displayed in the interface. Now leaders and followers can be determined:

\[
\begin{align*}
\text{leader}_{r,i} &= \arg\max(\rho_{\text{Exp}^{\text{early}}, \text{Exp}_{r,i}}); \\
\text{follower}_{r,i} &= \arg\max(\rho_{\text{Exp}_{r}, \text{exp}_i}^{\text{early}}).
\end{align*}
\] (4.6)

Note that there are none of the calculations used in SiMCAL 1 involving subtracting neighbor distances from succession distances. Again, this is due to rigorous testing with simulated data — it was found that these calculations added nothing to the accuracy of the analysis, and by omitting them in SiMCAL 3, the succession-finding process is considerably simplified. Of course, the succession-finding step is only performed if \( m > 1 \).

Clustering now proceeds exactly as in SiMCAL 2. Linking is done both as in SiMCAL 1, in which cluster succession relationships are determined (although in SiMCAL 3, these are based on entity successions rather than gene successions; a gene’s succession relationships are simply the union of its normal and abnormal entities’ succession relationships) and as in SiMCAL 2, in which cluster intersections are determined. Finally, gene differential expression determination is done as in SiMCAL 2.

Accuracies are comparable to those in SiMCAL 2 for gene differentiation, regardless of the values of \( n \) and \( m \). In addition, SiMCAL 3 finds succession relationships in simulated data with both sensitivity and specificity \( \approx 0.5 \) at \( r = 1 \), and specificity \( \geq 0.9 \) at higher ranks. The sensitivity of succession-finding at \( r > 1 \) naturally has \( \text{sens}_{r-1} \) as an upper bound; if a gene was found not to be
differentially expressed at $r - 1$, it is eliminated from consideration at $r$. However, genes involved in pathways marked out by succession relationships in the simulated data are usually correctly identified by the algorithm as differentially expressed at all but the highest ranks. Finally, $r_{\text{max}}$ remains low, as desired — $r_{\text{max}} = 6$ for the current “real” data set.

4.3 Interface and Performance Comparison

The reporting interface in SiMCAL 3, as might be expected, combines the features of those in SiMCAL 1 and 2. The look and feel is very much like that in SiMCAL 2, while gene and cluster visualization is like that in SiMCAL 1, with the added feature of showing both normal and abnormal expressions for genes (Fig. 4.1). Unlike either of its predecessors, the SiMCAL 3 interface is designed, as is the algorithm, to deal with data of any dimensionality.

The order of performance for SiMCAL 3 is equivalent to that of SiMCAL 2 with respect to $n$, although it is slightly slower overall due to the extra operations involved in succession-finding. SiMCAL 3 is also $O(m)$, which is unsurprising since the computational expense of finding $\Delta_1$ scales linearly with the size of the data.

4.4 Biological Interpretation

Among the more interesting differentially expressed genes in the current data set are:

100013.at Differentially expressed up to $r = 2$: a poorly documented gene which is, however, known to be involved in immune response.

100005.at Differentially expressed up to $r = 3$: a tumor necrosis factor (TNF) which plays a role in apoptosis.
Figure 4.1: A gene from simulated SiMCAL 3 data, $m = 8$, with its clusters.

100014_at Differentially expressed up to $r = 2$: a kinase involved in response to DNA damage.

100980_at Differentially expressed up to $r = 5$: an apoptosis signalling protein.

These genes, their leaders and followers, and other genes in their clusters may all be worthy of further and more intensive investigation.

Another gene of interest is 102914_s_at, a gene with wide-ranging effects on cell death, which is differentially expressed up to $r = 5$. Among its followers at high rank is 98869_g_at, a leukemia- and lymphoma-associated B cell protein involved in regulation of apoptosis; among its leaders is 99522_at, a cell cycle
signaller. These sorts of relations in the SiMCAL 3 analysis point toward the ability to “zero in” on pathways of profound biological and medical importance without the noise of genes which are not differentially expressed — a problem which plagued SiMCAL 1 and has now been corrected.
5. Conclusions and Future Directions

All three algorithms essentially meet their goals: they produce meaningful results while conforming to the SiMCAL specification. However, considerable future work can both improve the algorithms themselves and broaden their usefulness in bioinformatics research.

Future development on the family of algorithms presented here will almost certainly use SiMCAL 3 as a starting point, since it subsumes the functionality of SiMCAL 1 and 2. These earlier algorithms may now be viewed as prototypes — working prototypes to be sure, and capable of producing useful results on their own, but not adaptable enough to handle the wide range of data which is produced by modern microarray experiments. Future versions of SiMCAL 3 will be designated SiMCAL 3a, 3b, etc. as both the algorithm and the interface improve. Some goals include:

- Build a high-level interface which can not only show results, but also act as a controller for all aspects of the computational portion of the project. Preferably, the entire analysis process, from data import, through preprocessing, clustering, and linking, to reporting and visualization, should be available through the Web interface, so that users have a “one-stop” application.

- Perform statistical analysis on the operations of the algorithm to understand not only what it is doing, but why. In particular, it would be desirable to understand whether arbitrary calculations within the algorithm such as $\kappa$
are the best possible solution, or if more rigorous development can produce better solutions.

- Compare the results obtained from the algorithm with results obtained on identical data with different algorithms in more detail. Cross-validation would be a desirable outcome of this process.

- Obtain other data sets covering a variety of topics.

- Solicit user feedback on both the results of the algorithm and the interface.

The end users of the application are intended to be biologists, not mathematicians and computer scientists; although a number of biologists have been closely involved in the development process since the project’s inception, much more work is needed to make the application suitable for general use.

Looking farther ahead to SiMCAL 4, 5, etc., entirely different approaches to clustering may prove fruitful. Evolutionary search through the data space is one possibility; minimum spanning tree and other graph search algorithms present another. Nearest neighbor clustering is an intuitive, lightweight method, and one which can certainly produce good results, but it is far from the only avenue worth exploring in the SiMCAL context.

As a design principle, SiMCAL is a useful guideline for development in the ever-expanding field of microarray analysis. As a specific set of algorithms, it is a proven concept which has produced gratifying results. Its future lies in following the principles to produce a family of algorithms which build on the successes of those presented here.
REFERENCES


