Abstract

Data pipelining is the processing, analysis, and mining of large volumes of data through a branching network of computational steps. A data pipelining system consists of a collection of modular computational components and a network for streaming data between them. By defining a logical path for data through a network of computational components and configuring each component accordingly, a user can create a protocol to perform virtually any desired function with data and extract knowledge from them.

A set of data pipelines were constructed to explore the relationship between the biodegradability and structural properties of halogenated aliphatic compounds in a data set in which each compound has one degradation rate and nine structure-derived properties. After training, the data pipeline was able to calculate the degradation rates of new compounds with a relatively accurate rate.

A second set of data pipelines was generated to cluster new DNA sequences. The data pipelining technology was applied to identify a core sequence to represent a DNA cluster and construct the 95% confidence distance interval for the cluster. The result shows that 74% of the DNA sequences were correctly clustered and there was no false clustering.

APPLICATION OF DATA PIPELINING TECHNOLOGY IN CHEMINFORMATICS AND BIOINFORMATICS

Linyong Mao

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Master of Sciences
in the School of Informatics
Indiana University

December 2002

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Master of Sciences.

_____
Douglas G. Perry, Ph.D.

_____
Kenneth B. Lipkowitz, Ph.D.

Master's Thesis
Committee

_____
Sam A. Falk Milosevich, Ph.D.

Table of Contents

Chapter I. Introduction

I. Introduction to Bioinformatics

Over the last few decades, biotechnology has experienced a rapid development. Traditional biology, which studies the mechanism of life at the organism and cell level, has migrated to modern molecular biology, which studies life at the DNA, protein and other molecular levels. With the rapid progress of the human genome project, which provides a full spectrum of human genomic sequences, biological data are produced at an exponential rate. How to manage and analyse these data efficiently and effectively in order to convert them into the useful knowledge is a huge challenge. Enter bioinformatics, a new discipline that uses the mathematical modeling, statistical methods, and computational approaches to acquire, manage, analyze and visualize biological data.

The first journal dedicated to the research in bioinformatics, Computer Applications in the Biosciences, was published in 1985 [1]. Several years later, it changed its name to Bioinformatics [1]. In the later 1960's, Dayhoff created the first bioinformatics database, which included all the sequences available at that time [2]. The Protein DataBank, founded in 1972, had originally ten X-ray crystallographic protein structures. In 1987, the SWISS-PROT protein sequence database was established [2]. After that, GenBank was founded in the US and its first release was published in the form of printed book and computer tape [1]. Two sequence analysis tools, the Needleman-Wunsch and Smith-Waterman algorithms, were developed in 1970 and 1981, respectively [3].

Although databases were established early on, database searching tools evolved more slowly. At first, researchers used very simple methods such as keyword matching

to perform searches [2]. In 1990, a fast but not very sophisticated algorithm, BLAST, was used in searching databases, complemented with a sophisticated but slow algorithm, FASTA [2][3]. Since then, two important events transformed the entire bioinformatics field: rapid development of the internet and advances in sequencing technology [1]. The first of these advances made it possible for researchers to share data and cooperate world wide; the second generated huge quantities of genome data that must be analyzed and managed by computer [1].

Today, bioinformatics has become one of the hottest research areas and lots of scientists, including biologists, computer scientists, mathematicians and other disciplinary experts, have switched their research focus to bioinformatics. Accordingly, a large number of bioinformatics tools became available as a result of these scientists' efforts. These tools can be divided into the following categories according to their functions:

- *Databases*. Many databases exist and they can be further divided into nucleotide sequence databases, such as GenBank and EMBL, protein sequence databases, such as PDB and SWISS-PROT, sequence-structure databases, such as DSSP and DALI, sequence mapping databases, such as GeneMap 98 and Marshfield genetic map, and publication and bibliography databases, such as PubMed [4]. All of these databases can be accessed from the Internet and researchers can submit and retrieve sequences by using the software that is integrated into the database or by using other separate software. Among these databases, many are not isolated but instead, they are linked to each other.

- *Sequence alignment tools*. These tools are used to find homology regions in sequences. They can be used to perform pair-wise alignment or multiple sequence alignment. Most of the sequence databases have sequence alignment tools, such as BLAST and FASTA, so that users can do similarity searching. There are also some sequence alignment tools that users can install in their own machines, such as Clustal W. Sequence alignment tools can usually do both nucleotide and protein sequence alignment.
- *Predication tools based upon DNA sequences*. These tools are used to identify genes and other specific regions from DNA sequences. Some examples of these tools are GRAIL, which is used in gene prediction, FGENEH, which is used to find exons, and GENESCAN, which is used to find complete gene structures [5]. Again, some of these tools are built into the databases to facilitate the analysis.
- *Prediction tools based upon protein sequences*. These tools are used to predict the properties, structures and functions of proteins from their sequences. Most of these tools can also make predictions using DNA sequences. These tools include programs to predict the physical properties of proteins, such as SAPS and MOWSE; programs to determine the identity of a protein based on its amino acid composition, such as AACompIdent and PROPSEARCH; motif and pattern searching tools, such as BLOCKS and Pfam; secondary structure and folding class prediction tools, such as PREDATOR and nnpredict; special structure and feature prediction tools, such as COILS and PHDtopology; and tertiary structure prediction tools, such as UCLA-DOE and SWISS-MODEL [5].

- *EST analysis tools.* The expressed sequence tags (ESTs) are short fragments of genes. Although ESTs can be generated at a fast rate by using DNA microarrays, it is a time and labor consuming effort to analyze them. Currently, these sequences are stored in databases such as GenBank, EMBL and DDBJ [5]. These sequences can be used to discover genes and perform sequence polymorphism analysis, although the qualities of these sequences are currently not very good [5]. Tools such as UniGene can be used to classify them, and it is very easy and fast to find novel genes in dbEST [5].
- *Phylogenic analysis tools*. This kind of tool is used to find the evolutionary relationships among sequences. The basic steps for this kind of analysis include multiple sequence alignment, determining a substitution model, building a phylogenic tree, and evaluating that tree [5]. At present, the most widely used phylogenic analysis software is Phylip. There are also other tools such as PUZZLE and MOLPHY [5].
- *Comparative genome analysis tools*. Currently, the full range of several organisms' genomes have been mapped [5]. The genomes of these organisms can be compared to find the functions of unknown genes. There are several databases that can be used for this comparative genomic analysis, such as PEDANT and COGs. The database for a specific organism, such as for *E. Coli* and for *B. subtilis*, can also be used for such analysis [5].

Due to the limitations of the scope of this thesis, the above mentioned tools by no means provide a complete list of bioinformatics tools. Many scientists are still working

hard to develop the new tools and improve the quality of the available tools. With the completion of human genome project and other organisms' genome projects, large quantities of biological data are being generated. Instead of looking for weak similarities from scarce amounts of data, it now becomes possible and important to search for strong similarity from a wealth of data [1]. On the other hand, among genes discovered every day, many of them have similar functions and might be grouped into the same gene families, or they are probably the products of the proliferation of some genes within certain tissues. Thus, finding such information can eliminate data redundancy and reduce sample size [1].

The ultimate goal of bioinformatics is to fully understand the mechanism of life. To achieve this goal, we not only need to model how molecules function within a cell, but also model the intercellular interactions and understand how cells compose a tissue. After all these tasks have been accomplished, maybe we can begin to model the whole life system [1].

Although many bioinformatics tools exist, they almost always have some drawbacks. In the next section, challenges in bioinformatics will be discussed.

II. Challenges in Bioinformatics

Bioinformatics is an interdisciplinary approach to analyzing biological data. It involves expertise in biology, computer science, statistics, mathematics and other disciplines. Due to its interdisciplinary nature and the complexity of biological data, challenges arise in bioinformatics.

Database and information management

One important aspect of bioinformatics involves storage and management of a large amount of data. Challenges related to this subject include:

- *Data integration and data mining*. One of the goals of studying genes is to predict their functions. However, it is often impossible to accomplish this task either by just accessing a DNA database, a protein database, a protein 3-D structure database, or by only doing searches for similarities in sequence or structure. Integrating all the relevant information from related databases and using appropriate analysis tools are necessary to predict genes' functions. Currently, there are two ways to perform data integration. One is to embed external links to other databases within a database; the other is to integrate accesses across several data sources [6]. However, with so many existing databases, it is still a great challenge for data integration. Another important issue associated with information management is the data mining. Data mining is the procedure to discover the information that is implicit--previously unknown yet useful--from existing data. Widely used data mining methods include machine learning, statistical methods, etc. With the large amounts of data out there but yet to be analyzed, and, with new raw data generated every day, the development of efficient data mining methods and their application become urgent.
- *Scalability*. In April 2001, there were 300 complete genomes, 15,000 macromolecular structures, 400,000 protein sequences, 11.5 million DNA sequences, and 11 million citations related to bioinformatics [6]. And these numbers are increasing at a fast rate. On average, the size of the GenBank

database doubles every 15 months. Therefore, the scalability of bioinformatics tools needs to be considered at the designing phase of these tools [6].

- *Redundancy and multiplicity of data*. All biological data can be grouped according to similarities of their biological meaning to avoid redundancy and multiplicity [6]. For example, genes can be grouped according to their functions; proteins can be grouped by their structures. This kind of grouping simplifies databases. For example, the number of gene sequences is very large, however, the number of the structures of the proteins they encode is relatively small.
- *Data standardization*. There are many public and private databases and many of them have their own data definitions and formats. In addition, different databases have different naming rules for the same data. Although some effort has been made to solve some of these problems (e.g., most databases support FASTA format and use a GenBank sequence ID), more work needs to be done to establish universal naming and formatting systems.

Homology search

One way to find the function of a new gene is to find homologies between the gene of interest and other genes whose functions are known. Many programs have been developed to do homology searches. One of the major challenges is how to search for remote similarities [7]. In this case, it is very difficult to extract signals from noise [6]. Another challenge is how to improve the efficiency of algorithms [7]. Choosing a model that is appropriate for a particular domain is also one of the research areas in homology searching [6].

Sequence multiple alignment

Many sequence multiple alignment programs, such as Clustal W, don't guarantee finding optimal solutions [5]. Users have to carefully choose the parameters and provide their expert knowledge to find solutions that have true biological meaning [5]. How these sequence multiple alignment tools can be designed to have some intelligence so that they can perform these tasks automatically remains a challenge.

Protein structure and function prediction

Although several predictive tools using protein sequences have been successfully used in some studies, the overall accuracy rate of prediction is not very high. For example, PredictProtein has an average accuracy rate of 72% and nnpredict has an accuracy rate of 65% [5]. Both tools are used to predict protein secondary structure. It is known that the primary structure and some environmental factors determine the 3-D structure of a protein. But the mechanism governing this determination is not very clear. For algorithms to predict protein structures, the complexity of the search space is still a challenge for software developers [7].

Automated text analysis

It takes a lot of time and effort to manually extract useful information from a textual database, such as Medline, even for an expert [8]. Some relations among several different objects studied in different publications, such as a relation between a protein and a disease, can sometimes be found. Due to the size of the textual database, automated

text analysis tools are needed to facilitate the extraction of useful information from the literature [8]. Although there are some techniques to solve this problem, such as natural language processing (NLP), overall performance is not satisfactory [8].

III. Introduction to Chemical Informatics

With the invention of new technologies such as combinatorial chemistry and high throughput screening, the amount of information in chemistry is increasing dramatically. This is evident from the fact that the Chemical Abstracts Service adds over 700,000 new compounds to its database annually [9]. Each new compound has its own physical and chemical properties (e.g., reaction information) that need to be stored in the database. Chemists have developed a nomenclature system to name a substance that adds another dimension to chemical information. The development of an informatics infrastructure with the capability not only to acquire and manage huge amounts of data but also to discover the relations, trends and patterns from the data is the major concern of chemical informatics.

Studies in chemical informatics include molecular simulation, chemical information management, and data analysis techniques with high quality graphical visualization. Chemical visualization and modeling techniques are revolutionizing chemical research. A novel compound can be built on a computer and its physical and chemical properties and its interactions with other molecules, can be calculated with a high degree of accuracy using a molecular modeling package even before the compound is actually synthesized.

Chemical informatics techniques are used extensively by the pharmaceutical industry to facilitate and accelerate the drug discovery process. Chemical informatics also plays an important role in instrument design. With the inclusion of modern sensors in chemical instrumentation, a large volume of data is generated. Future instrumentation needs to include the ability to read data from existing chemical databases and have the ability to analyze data as it is generated with the modeling technique. Such abilities will enable instruments to make intelligent decisions while the data is being collected and analyzed. The incorporation of chemical informatics will provide a competitive advantage to the companies that are involved in medical, environmental, and chemical instrumentation research.

Research areas in chemical informatics include [10]:

- Molecular modeling
- Chemical database systems (including spectral and reaction systems)
- Chemometrics (the use of mathematical, statistical, and other methods of formal logic to determine, by indirect means, properties of substances that otherwise would be very difficult to measure)
- Automated synthesis
- High throughput screening
- Structure coding systems (including nomenclature)
- Electronic chemical publishing systems
- Chemical patent information sources
- Laboratory automation
- Laboratory information management systems (LIMS)

IV. Challenges in Chemical Informatics

Many people view chemical informatics as an extension of chemical information, which is a well established concept covering many areas that employ chemical structures, data storage, such as compound library and online chemical literature, and computational methods, such as SAR (structure-activity relationship) analysis and molecular property calculation [11]. In this section, some challenges faced by researchers in the chemical informatics field are discussed.

Bridging the gap between bioinformatics and cheminformatics

In the field of life science research, huge amounts of data are generated by genomics, proteomics, high throughput screening (HTS), and combinatorial chemistry. While cheminformatics scientists have focused on HTS and chemical data, bioinformatics scientists have focused on genomic and proteomic data. There is typically little or no interaction between these groups [12]. However, it is important for cheminformatics scientists and bioinformatics scientists to communicate with each other constantly and to make effective use of data from each other in order to find scientific relationships and build hypotheses. For example, drug discovery today involves both bioinformatics for target discovery and cheminformatics for lead identification [13]. If the data from these two fields can be integrated, the task of achieving new insights and discovery of hidden relationships may become easier. Providing the technologies to bridge these islands of information and allow researchers to query across diverse data domains is one challenge faced by informaticists.

Chemical structure representation

Currently, different data sources store and represent chemical structures in a variety of proprietary file formats, making it difficult to access and interchange them among applications [14]. The Life Sciences Research/Cheminformatics Task Group of the Object Management Group is currently developing the chemical structure access and representation standard. It is an open standard based on XML/CML (extensible markup language/chemical markup language), an open W3C data representation standard [15].

Challenges in molecular modeling

The three most important problems in molecular modeling include [16]:

- The difficulty of calculating free energies of molecules by computer.
- Representation of solvent effects in calculations.
- Simulation of chemical reactions.

Given the advances in theory, hardware, and software one expects that these issues will become less problematic in the near future.

References

1. Higgins, D.; Taylor, W. *Bioinformatics: Sequence, Structure and Databanks.* Oxford: New York, 2000.

2. Gardner, S. The Evolution of Bioinformatics. Retrieved June 15, 2002, from http://www.bitsjournal.com/sgard.html.

3. Richon A. B. A Short History of Bioinformatics. Retrieved June 15, 2002, from http://www.netsci.org/Science/Bioinform/feature06.html.

4. European Bioinformatics Institute: http://www.ebi.ac.uk/, 2002.

5. *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*. Baxevanis, A. D.; Francis Ouellette, B. F., Ed.; John Wiley & Sons: Denver, 2001.

6. Kaminski, N. *American Journal of Respiratory Cell and Molecular Biology* **2000,** *23(6)*, 750-711.

7. *Computational Methods in Molecular Biology.* Salzberg, S. L.; Searls, D. B.; Kasif, S.; Eds.; Elsevier: Amsterdam, Netherlands, 1999.

8. Stephens, M.; Palakal, M.; Mukhopadhyay, S.; Raje. R. Detecting gene relations from Medline abstracts. Pacific symposium on biocomputing, 2001.

9. Chemical Abstracts Service: http://www.cas.org, 2002.

10. Indiana University School of Informatics: http://www.informatics.iupui.edu/, 2002.

11. Ritchie, T. Chemoinformatics: manipulating chemical information to facilitate decision- making in drug discovery. *Drug Discovery Today* **2001,** *6(16)*, 813-814.

12. Langton, W.; Higgins, M. Mind the gap: Bridging the gulf between bioinformatics and cheminformatics, 221st ACS National Meeting, San Diego, CA, April 1-5, 2001.

13. Tozer, J. R. Ask More of Your Discovery Data. *Genomics & Proteomics* **July/August 2001**, 65-68.

14. Cambridge Healthtech Institute: http://www.genomicglossaries.com/content/chemoinformatics_gloss.asp, 2002.

15. Fry, J. An Open Chemical Structure Interchange Form Based on WC3 Standards, Cambridge Healthtech Institute's 6th Annual Cheminformatics Meeting, Philadelphia, PA, May 6- 8, 2002.

16. Leach, A. R. *Molecular Modeling*; Addison Wesley Longman: Singapore, 1996; Chapter 9.

Chapter II. Using Data Pipelining to Convert Data into Useful Knowledge

I. Introduction

With huge volumes of raw data being generated by new advances in bioinformatics and chemical informatics, a new technology, data pipelining, has been developed to convert data into useful knowledge by addressing the challenges of scalability, data integration and data mining.

Data pipelining is the processing, analysis, and mining of large volumes of data through a branching network of computational steps [1]. A data pipelining system consists of a collection of modular computational components and a network for streaming data between them. Some components make simple calculations, like standard deviation, some components merge or sort data according to key properties in the data, some filter or branch data to different downstream components based upon data properties or live calculations, and some components function to simply import or read a particular source of data [2]. By defining a logical path for data through a network of these components and configuring each component accordingly, a user can create a protocol to perform virtually any desired function with data, and subsequently extract knowledge.

Data pipelining technology serves as the complement to database systems for discovery informatics. Data pipelining does not store or manage data collections itself, it imports data collections from flat files or databases and stores the running results in a file or a database. The nature of database systems requires that research inquiries be restricted to what has been pre-conceived and already calculated in the system [3]. With

the technology of data pipelining, it is possible to mine data from multiple disparate collections without the need to unify the data in a single database. Database technology, along with data pipelining, provides uncompromising management and analysis capabilities.

In the following sections, we will discuss the desired functional requirements, system requirements, overall design, manuals, screen layout and report layout of a well-designed data pipelining application.

II. Functional Requirements of Data Pipelining

The data pipeline itself must have extremely high performance if it is to keep pace with the rate of data generation. Modern discovery research relies upon many different data sources across entire scientific domains, and in many cases it is their collective analyses that provide the most valuable insights. The ability to integrate data from disparate sources is an essential requirement of data pipelining. Data from multiple disparate sources is fed into a single processing pipeline. By defining the relationships between data in the pipeline itself, the sources of data themselves need not be altered in any way.

Component-based Pipeline Definition

Pipelines should be viewed conceptually as collections of modular computational components within a framework for streaming data between them [4]. A pipeline for a specific task can be created by selecting appropriate components, configuring each component accordingly, and defining a logical path for streaming data through these components.

Modularity of Components

A data pipeline consists of connected computational components. The individual component must be modular to allow its seamless insertion into any new protocol. In order to allow data to run through components in a pipeline, one component should be compatible with the other components.

Data Modeling Capability

Some computational components can be implemented with artificial intelligence methods, machine learning techniques, statistical methods, etc. to identify trends, discover unusual patterns, and find hidden relationships from multiple disparate data sources. This learning capability greatly extends the function of data pipelining.

Open Architecture

With the rapid progress of scientific research, new ideas and new research fields are emerging. On the case when none of the present components provides a desired functionality, the modular design and open architecture of the system should allow a user to create new components for his proprietary purposes.

Pipeline Protocols

A data pipeline protocol is a logical layout of modular components to perform a specific task. Once created, these protocols can be saved for future re-use or to be shared with a broad community of users.

Visualization and Reporting

Visualization functionality should be provided by the system to allow a user to design new pipelines, monitor the status of pipeline executions, and save pipeline

protocols. In addition, a user will desire the visualization and reporting functionality to view results and present them to others.

III. System Requirement and Human Resource

A data pipelining system adopts the client/server architecture. The Unix server stores databases (e.g., in Oracle format) that can be queried and the query results are then fed into a data pipeline. A user can access the server through a web interface on a client machine. The web interface provides available computational components and tools to construct a pipeline. After a pipeline is constructed, the job is submitted to the server and run on the server. The execution results can be either saved as a file on a client machine or loaded into the database that is stored on the server.

Human Resource Requirement

- Project Manager: this person is responsible for the installation, system maintenance, interacting with users, etc.
- Unix System Administrator: a person needed to install, configure and maintain servers and storage devices since our databases are stored on a Unix system.
- Oracle Database Administrator: a person who is responsible for installing the Oracle database and maintaining it.
- End Users: the bioinformatics and cheminformatics scientists.

IV. Overall Design

Since a data pipeline is constructed from modular computational components, exactly which components should be provided and how each of these components should

be designed become critical issues. The following types of components are usually included in the data pipelining system:

Data Reading Components

In order to integrate different data sources, many components function to simply import or read a particular source of data. Data reading components are a group of components, each of which reads or imports an Oracle database, delimited flat files, and SD molecule records, etc.

Calculation Component

This component can make simple calculations, like standard deviation, or perform live calculations on the available data source to derive new properties.

Data Merging Component

This component merges data from different sources, finds duplicates, and branches data to different downstream components.

Data Sorting Component

This component sorts data records in a file according to a specific file of the data record.

Data Filtering Component

This component filters data to different downstream components based upon data properties.

Data Modeling Component

This component is implemented with some artificial intelligence method. It extends the capabilities of data pipelining by automatically learning from the data. The learning ability is realized with a straightforward learn-by-example paradigm: users

simply mark sample data that have the traits they are looking for, and the modeling component learns to distinguish it from other background data. The data modeling component automatically identifies the properties that can affect the trait and weights them accordingly.

Results Viewing Component

This component enables a user to view the execution results of data pipelining with the visualization tools.

Data Writing Component

This component gives a user the option to either save the running result of data pipelining in the format of a flat file on a client machine or load the result into the database stored on the server.

Molecular Modeling Component

This component is designed for tasks related to cheminformatics. It reads a molecular structure, and calculates molecular properties based on its structure using molecular mechanics, semi-empirical, or quantum mechanics methods.

Molecular Fingerprint

This component is designed for tasks related to cheminformatics. It generates a fingerprint (a binary bit string) to represent the three dimensional structure of a molecule.

Bioinformatics Component

This is a group of components, each of which is dedicated to a specific function. It typically includes components for:

- Homology searching: sequence similarity and homology search against a nucleotide or a protein database

- Multiple sequence alignment
- Gene prediction
- DNA sequence translation
- Protein motif and pattern searching
- Protein fold classification and structural alignment
- Protein secondary and tertiary structure prediction
- Protein function prediction

After different types of components become available, a user can construct a data pipeline for a specific task by selecting an appropriate set of components, configuring each of them accordingly, and connecting these components to allow data to run through them.

V. Manuals, Screen Layout and Report Layout

The first step typically involves a user to open Internet Explorer or Netscape to connect to the data pipelining server. After the user provides his login name and password, an interface I designed for the data pipelining system (Figure 2.1) is presented to the user to allow him to build his data pipeline and submit the job to the server.

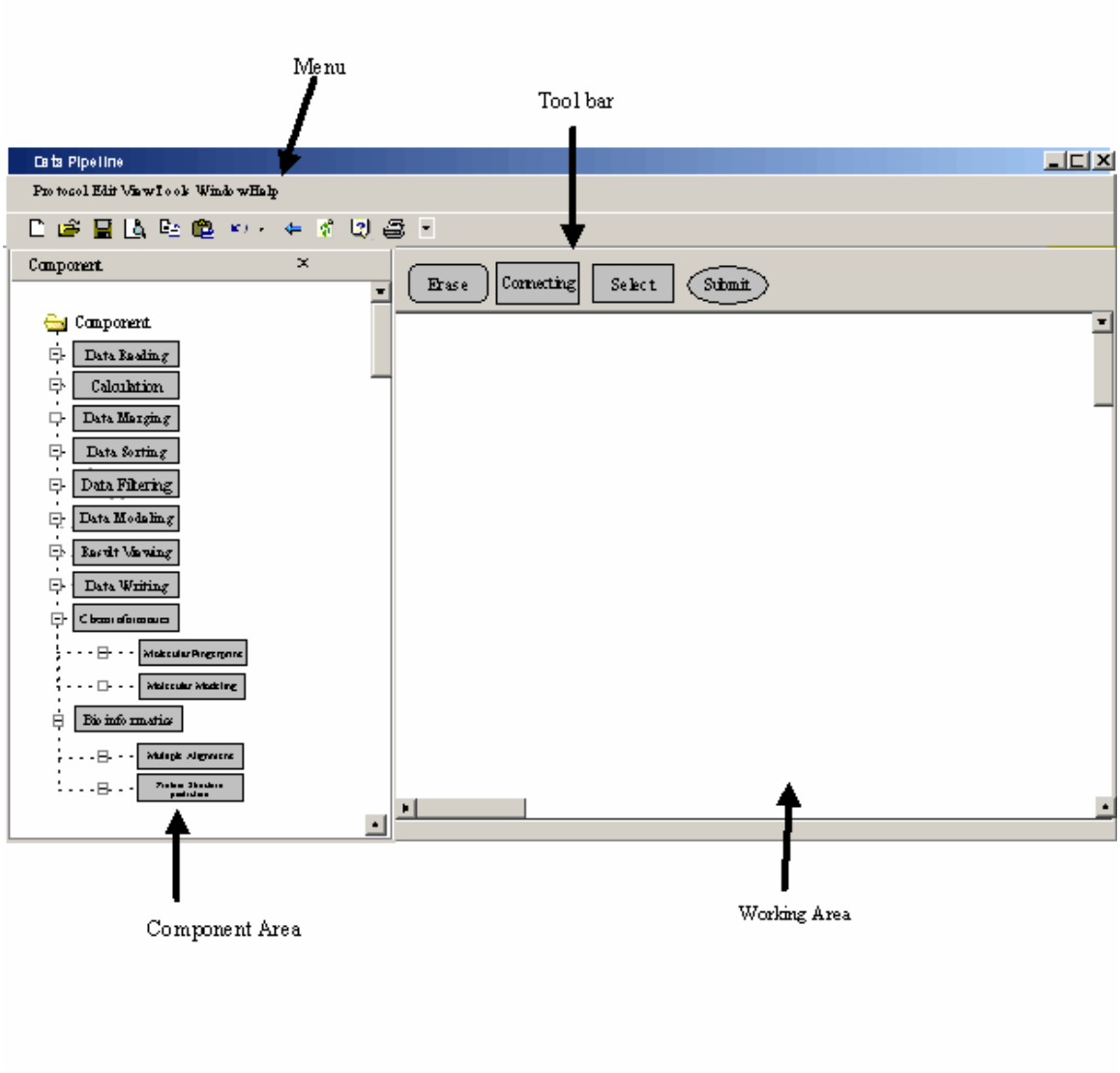


Figure 2.1 A data pipeline web interface.

The interface consists of a menu, component area, tool bar and working area:

- The menu includes Protocol and Edit. When the Protocol button is clicked, a menu pops out which contains Protocol Templates, Read Protocols, Save Protocols, Save as, and Print Protocols. Protocol Templates provide a user with a collection of popular, routinely used protocols. Read Protocols allows a user to import a saved protocol into the working area. Save Protocols permits a user to

save the current protocol constructed in the working area on the local PC machine. Print Protocols prints out the protocols showing in the working area. Edit menu contains Redo, Undo, Select, Copy, Paste and Erase. Select allows a user to select a particular component in a pipeline, Copy allows a user to paste the pipeline image into a word document or other formats of documents, and Erase allows a user to remove the selected component from a pipeline.

- The component area contains all types of components described in the Overall Design section.
- The tool bar contains Connecting, Select, Erase, and Submit. Connecting allows a user to bridge two components, A and B, thus output from the component A can be directed to the component B as input. The Select and Erase are the same as those contained in the Edit Menu. When the Submit button is clicked, execution of a constructed data pipeline will be submitted to the server.
- The working area is where a user builds and edits a data pipeline.

A user selects a component by clicking that component in the component area and then clicks it again in the working area. The component now appears in the working area. By right-clicking the component in the working area, a configuration window pops out that allows a user to modify parameters for that component. For example, a user can specify the name of a flat file and the delimiter used in the file for a data reading component to retrieve data records in that file. Or, a user can issue an SQL (structural query language) statement in the configuration window of a data reading component to retrieve data records from a database. A user configures a data sorting component by specifying a particular data field as the sorting criterion. A filtering component is

configured by setting the filtering criteria that may be a range of a property such as molecular weight. A data writing component can be configured to save the running result of data pipelining either in the format of a flat file on a client PC or in a database stored on the server.

A component has one or more input lines and/or one or more output lines. For example, a data merging component has more than one input line and more than one output line, as demonstrated in figure 2.2.

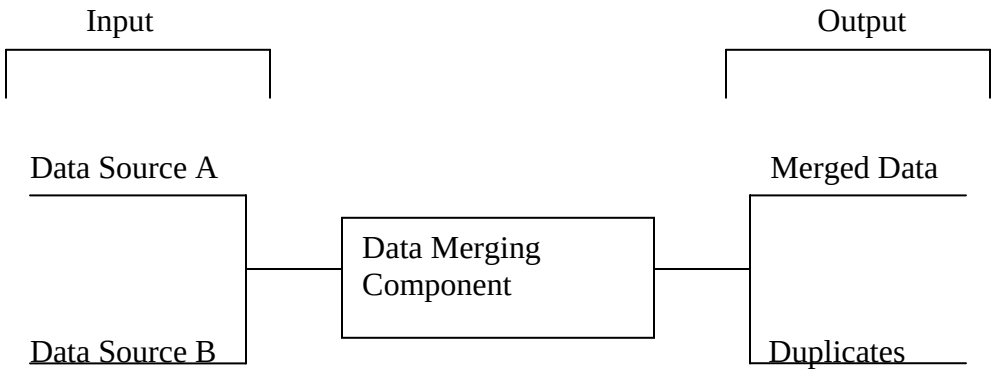


Figure 2.2 The input lines and output lines of a data merging component.

How then do we connect components to allow data to run through them? The example in figure 2.3 illustrates how to make connections between components.

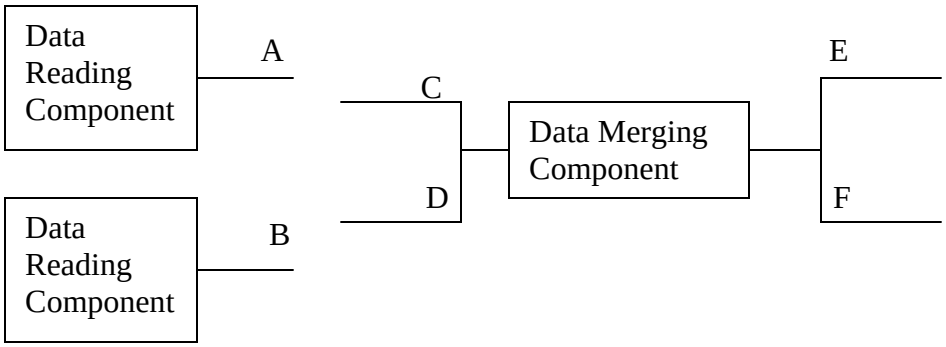


Figure 2.3 Three separate components appearing in the working area.

The above figure shows three separated components in the working area: two data reading components and a data merging component. To connect them, a user clicks the Connecting button on the tool bar. The program prompts the user to first select an output line and then an input line. For example, in the figure above, output line A and input line C are selected sequentially. After C is selected, a line connecting A and C appears in the working area indicating that data flows from A to C. Similarly, data flowing from B to D can be made. Three connected components are shown in figure 2.4.

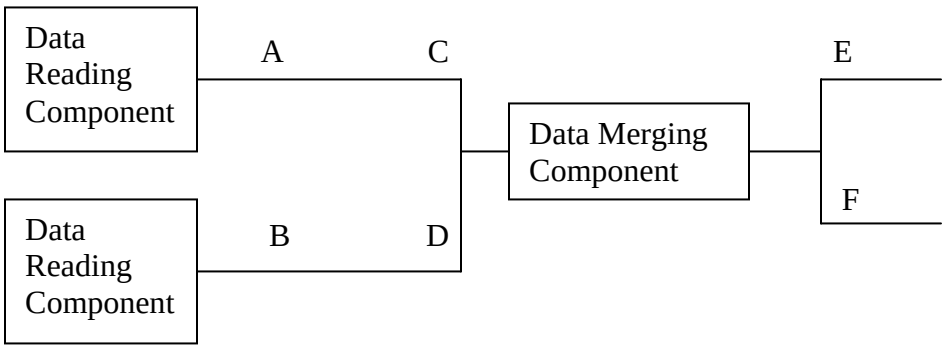


Figure 2.4 Illustration of making connections between components.

In Figure 2.5, line E outputs the merged data and directs these data to a data writing component. Similarly, line F outputs the duplicates to be shown in the result viewing component.

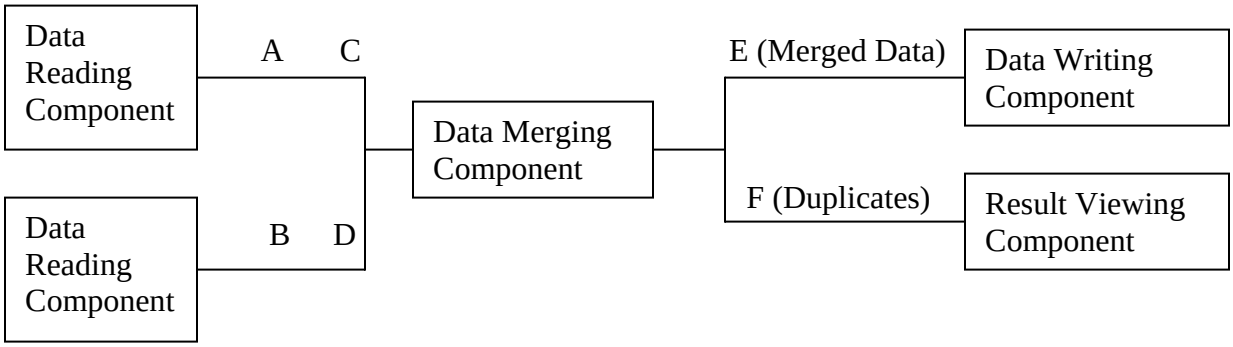


Figure 2.5 An example of a data merging protocol.

A data pipeline in the working area can be edited using the Select and Erase tools in the tool bar. Once a pipeline is constructed, a user can click the submit button in the tool bar to submit the job to the server, and the execution of a data pipeline will run on the server. The server will inform the user after the job is completed. A constructed data pipeline can be saved in the local machine for the future re-use or edit.

Report layout is protocol specific. Different protocols have different report layouts. The details of report layout will be demonstrated in the following chapter.

References

1. Scitegic: http://www.scitegic.com/main.html, 2002.

2. Tozer, J. R. Ask More of Your Discovery Data. *Genomics & Proteomics* **July/August 2001**, 65-68.

3. Kostrowicki, J.; Peng, Z.; Kuki, A. A River of Data Runs Through It. *Modern Drug Discovery* **2001**, *4(5)*, 94-96.

4. Resnick, R. (December, 1999) White Paper: A Scalable, Extensible High-throughput Pipeline Framework. http://www.bitsjournal.com/informatics_genomics.html.

Chapter III. Application of Data Pipelining Technology in Cheminformatics and Bioinformatics

I. Establishing Quantitative Structure Biodegradability Relationship (QSBR) Using Data Pipelining

In chemistry, the activity of a molecule is often a composite of many factors. A structure-activity study can help to identify which features of a molecule give rise to its activity and help to make modified compounds with enhanced activities. A quantitative structure-activity relationship (QSAR) relates numerical properties of the molecular structure to the activity via a mathematical model [1].

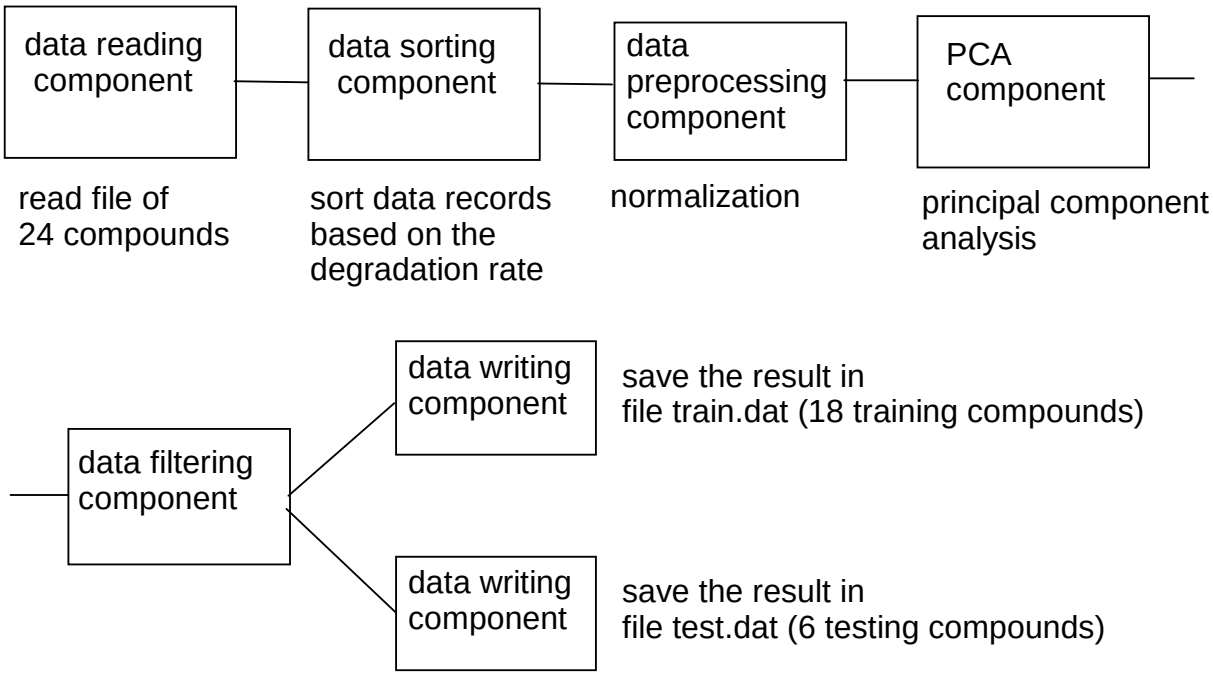
The table 3.1 lists a typical QSAR data set [2]. There are 24 compounds in the data set. For each compound, one activity (Y2) was determined experimentally and nine structure-derived properties (Mw through BCLU) were calculated using molecular modeling packages [2]. Y2 represents the dehalogenation rate constant obtained from assays using the intact cells of *Rhodococcus erythropolis* Y2. It is expressed as a percentage of the dehalogenation rate constant obtained under the same conditions with 1-chlorobutane. Nine structural properties used to describe the hydrophobicity, steric and electronic characteristics of each compound are: molecular weight (Mw), moments of inertia along the x-axis (IX), logarithm of the octanol/water partition coefficient (logP), heat of formation (Hf), total energy (TE), electronic energy (EE), energy of the highest occupied molecular orbital (HOMO), dipole moment (Dip) and the bond contribution of the lowest unoccupied molecular orbital (BCLU).

Table 3.1. Dehalogenation rates and descriptors [2]

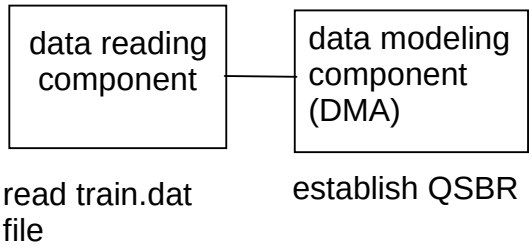
	Y2	Mw	IX	logP	Hf	TE	EE	HOMO	Dip	BCLU
1-chlorobutane	100	92.57	4.97	2.64	-39.82	-1010.99	-3438.47	-11.1327	1.74	-1.98
1-chloropentane	105	106.60	5.67	3.05	-46.67	-1166.82	-4424.73	-11.1326	1.76	-1.97
1-chlorohexane	99	120.62	7.18	3.58	-53.51	-1322.66	-5475.49	-11.1325	1.77	-1.98
1-chloroheptane	87	134.65	7.94	4.15	-60.36	-1478.49	-6581.83	-11.1329	1.78	-1.98
1-chlorooctane	62	148.68	9.37	4.64	-67.20	-1634.33	-7737.48	-11.1329	1.78	-1.98
1-chlorononane	51	162.71	10.16	5.17	-74.05	-1790.16	-8937.12	-11.1326	1.79	-1.98
1-chlorodecane	38	176.73	11.51	5.70	-80.90	-1946.00	-10176.60	-11.1327	1.79	-1.98
1-chlorododecane	20	204.79	13.66	6.76	-94.58	-2257.66	-12762.00	-11.1021	1.79	-1.98
1-chlorotetradecane	16	232.84	15.84	7.81	-108.28	-2569.33	-15471.70	-11.0619	1.80	-1.98
1-chlorohexadecane	0	260.85	17.87	8.87	-121.98	-2881.00	-18288.40	-11.0270	1.80	-1.98
1-chlorooctadecane	0	288.95	25.05	9.93	-135.53	-3192.67	-21210.10	-10.9987	1.80	-1.98
1-bromoethane	92	108.97	2.71	1.61	-13.12	-678.77	-1658.57	-10.6925	1.66	-1.99
1-bromobutane	108	137.03	5.18	2.75	-26.77	-990.44	-3381.10	-10.6882	1.72	-2.00
1-bromohexane	78	165.08	7.56	3.80	-40.45	-1302.11	-5414.47	-10.6878	1.75	-2.00
1-bromotetradecane	27	277.30	16.61	7.95	-95.24	-2548.79	-15405.20	-10.6901	1.77	-2.00
1-iodobutane	73	184.02	5.39	3.05	-14.69	-984.41	-3348.76	-10.4276	1.56	-1.99
1-iodopentane	65	198.05	5.98	3.58	-21.54	-1140.25	-4331.49	-10.4277	1.58	-1.99
1-iodohexane	35	212.08	7.93	4.11	-28.38	-1296.08	-5379.64	-10.4277	1.58	-1.99
1,3-dichloropropane	152	112.99	5.12	2.00	-40.75	-1215.29	-3511.31	-11.3721	1.51	-2.02
1,4-dichlorobutane	155	127.01	5.34	2.24	-48.09	-1371.15	-4497.10	-11.2981	0.00	-2.00
1,6-dichlorohexane	113	155.07	7.69	3.29	-62.02	-1682.83	-6653.46	-11.2219	0.00	-1.99
1,9-dichlorononane	66	197.15	11.99	4.88	-82.66	-2150.33	-10248.00	-11.1794	1.54	-1.98
1,10-dichlorodecane	60	211.18	12.29	5.41	-89.53	-2306.17	-11523.90	-11.1707	0.05	-1.98
1,2-dibromoethane	87	187.87	11.28	1.96	-7.56	-1018.35	-2522.43	-10.7587	2.21	-1.99

Four data pipelines, as shown in figure 3.1, were constructed in the working area of a web interface, which was described in Chapter 2, to establish the Quantitative

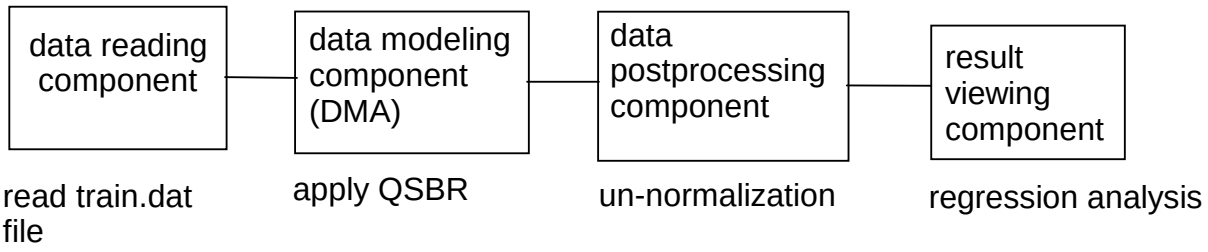
Structure Biodegradability Relationship (QSBR) and apply it to predict the dehalogenation rates.



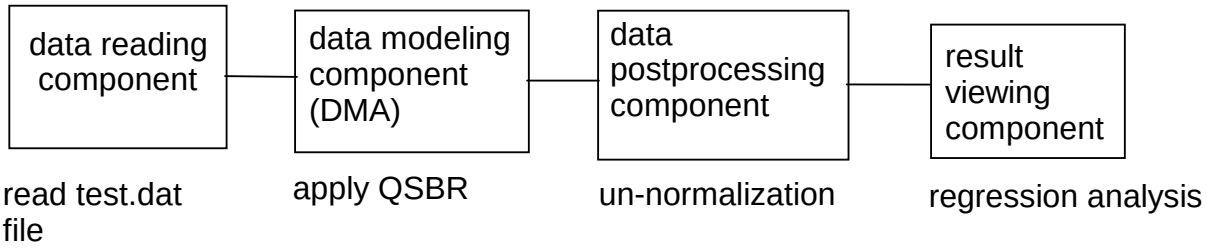
Pipeline A: Dividing the transformed data set (24 compounds) into a training set (18 compounds) and a testing set (6 compounds).



Pipeline B: Using a neural network to establish structure-biodegradability relationship.



Pipeline C: Applying the QSBR model to calculate the biodegradation rates of 18 training compounds, and comparing them with the actual values.



Pipeline D: Applying the QSBR model to calculate the biodegradation rates of 6 testing compounds, and comparing them with the actual values.

Figure 3.1 Four data pipelines (A-D) constructed in the working area of a web interface.

Pipeline A first read 24 compounds and sorted the data set according to the Y2 values. After the sorted data set ran through the data preprocessing component, both original inputs (9 structural properties) and original targets (Y2) were normalized so that both of them had zero means and unity standard deviations in order to derive the structure-biodegradability relationship more efficiently. The normalized inputs were further processed by the PCA (principal component analysis) component. The PCA technique had three effects on the normalized inputs: it orthogonalized the components of the input vectors so that they became uncorrelated with each other; it sorted the resulting orthogonal components (principal components) based upon their contributions to the total variation in the data set; and it eliminated those components that contributed the least. In

the data pipeline A, the PCA component was configured such that the principal components with less than 5% contributions to the total variation in the inputs were discarded. The result was that only the top four principal components remained. The normalized value of Y2 and the top four principal components for each of the 24 compounds are listed in table 3.2.

Table 3.2 The normalized value of Y2
and the top four principal components for each of the 24 compounds

	Normalized Y2	1st principal component	2nd principal component	3rd principal component	4th principal component
1-chlorohexadecane	-1.6624	-4.3603	0.2807	-0.0686	0.1320
1-chlorooctadecane	-1.6624	-5.8564	0.5591	0.2044	0.1845
1-chlorotetradecane	-1.2844	-3.2739	0.0796	-0.3087	0.1705
1-chlorododecane	-1.1899	-2.1867	-0.1465	-0.5429	0.2054
1-bromotetradecane	-1.0246	-3.0549	1.5713	1.5686	0.5397
1-iodohexane	-0.8356	0.9849	1.8684	0.5988	-0.8869
1-chlorodecane	-0.7648	-1.1077	-0.3398	-0.7794	0.2376
1-chlorononane	-0.4577	-0.5463	-0.4024	-0.8898	0.2267
1,10-dichlorodecane	-0.2451	-1.6550	-1.8926	1.0809	-1.5312
1-chlorooctane	-0.1978	-0.0309	-0.4651	-0.9869	0.2155
1-iodopentane	-0.1270	1.5780	1.7942	0.4860	-0.9055
1,9-dichlorononane	-0.1033	-1.2769	-0.6246	-0.4284	-0.0062
1-iodobutane	0.0620	2.0616	1.7267	0.4019	-0.9225
1-bromohexane	0.1801	1.2925	1.0912	0.7160	0.4580
1-chloroheptane	0.3927	0.5230	-0.5281	-1.0973	0.2046
1,2-dibromoethane	0.3927	1.7543	1.5962	-0.3397	0.2714
1-bromoethane	0.5108	3.2634	0.6465	-0.3193	-0.2271
1-chlorohexane	0.6762	1.0362	-0.5902	-1.1927	0.1932
1-chlorobutane	0.6998	2.0666	-0.7307	-1.3786	0.1539
1-chloropentane	0.8179	1.4412	-0.7701	-1.9834	-0.4048
1-bromobutane	0.8888	2.3555	0.9426	0.5293	0.4156
1,6-dichlorohexane	1.0069	0.6655	-2.1992	1.3520	-0.9588
1,3-dichloropropane	1.9281	2.4581	-1.0610	1.5967	2.5169
1,4-dichlorobutane	1.9990	1.8681	-2.4061	1.7811	-0.2827

The data filtering component divided the transformed dataset into a training subset of 18 compounds and a testing subset of the remaining 6 compounds whose Y2 values ranked 2nd, 6th, 10th, …, 22nd in Table 3.2. The file *train.dat* and file *test.dat* had 18 training compounds and 6 testing compounds respectively.

Pipeline B, C and D were submitted to the data pipeline server at the same time for execution. Pipeline B was built to generate a predictive model of dehalogenation rates. It first read the file *train.dat*. This file had 18 compounds. For each compound, it had one normalized value of dehalogenation rate constant and four principal components representing its structural properties. The data modeling component was implemented with a two-layer backpropagation neural network. The neural network was configured with two neurons in the hidden layer and one neuron in the output layer, as indicated in figure 3.2. The Tan-Sigmoid transfer function[1] was used for the hidden neurons and a linear transfer function (y = x) for the output layer. A 4-element input was fed into the neural network, corresponding to four principal components. The Levenberg Marquardt (LM) algorithm[2] was applied to train the neural network to overcome the slow convergence caused by the gradient descent training algorithm. The LM algorithm appears to be the fastest method for training backpropagation networks with up to a few hundred weights to perform function approximations [3]. The advantage becomes even more pronounced if very accurate training is desired. In many cases, the LM algorithm is able to achieve lower mean square errors than any of the other algorithms tested [3]. The network was trained using 18 training compounds to establish a relationship between molecular structure and the dehalogenation rate. The network was trained in a batch mode, which updated the weights and biases after all of training data set was applied to the network. The total training iterations were set to 500 epochs, and the performance

[1] Tan-Sigmoid transfer function: y = [1-exp(-x)] / [1+exp(-x)] It squashes the infinite input range, (-∞, +∞), into the range of (-1, +1).
[2] LM algorithm: A neural network training algorithm that was designed to approach second-order training speed without having to compute the Hessian matrix.

goal was set to zero. The data modeling component was given a particular name like DMA in its configuration window.

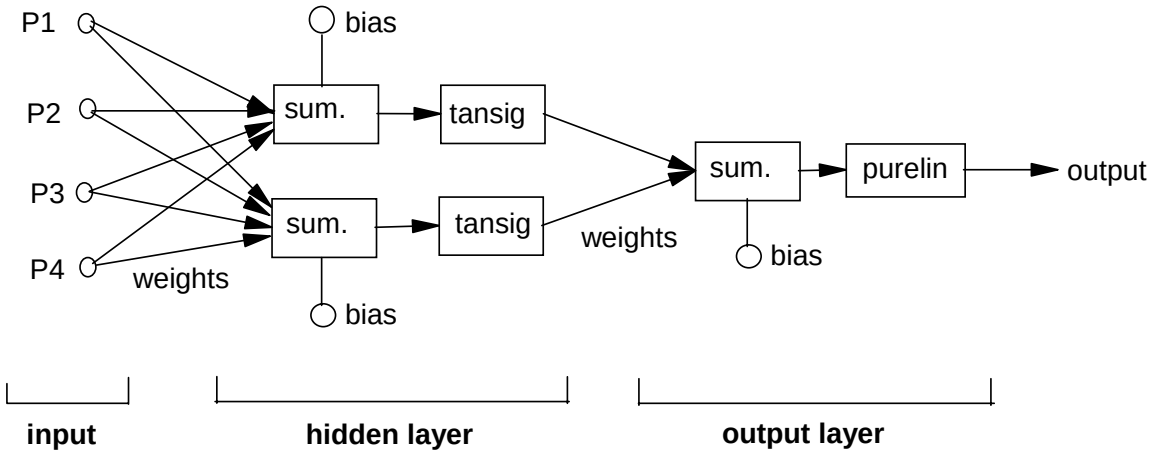


Figure 3.2. The architecture of the neural network created to implement the data modeling component in pipeline B.

After the QSBR model was developed in the pipeline B, the model was applied to calculate the dehalogenation rates of 18 training compounds, as shown in the pipeline C (Figure 3.1). After four principal components of each compound were fed into the data modeling component DMA as an input to the neural network, DMA computed its dehalogenation rate in the normalized form, which was un-normalized by the data postprocessing component. The result viewing component applied linear regression analysis to compare the output from DMA (A) with the actual dehalogenation rates (T), as shown in Figure 3.3. The dashed line in the figure indicates a perfect fit: A = T. The solid line shows the actual linear relation between the DMA outputs and actual dehalogenation rates: A = 0.989T + 0.821, which is very close to the perfect fit. The

correlation coefficient (R), 0.994, indicates that 99.4% of the variation in the DMA outputs is explained by the variation in the actual values.

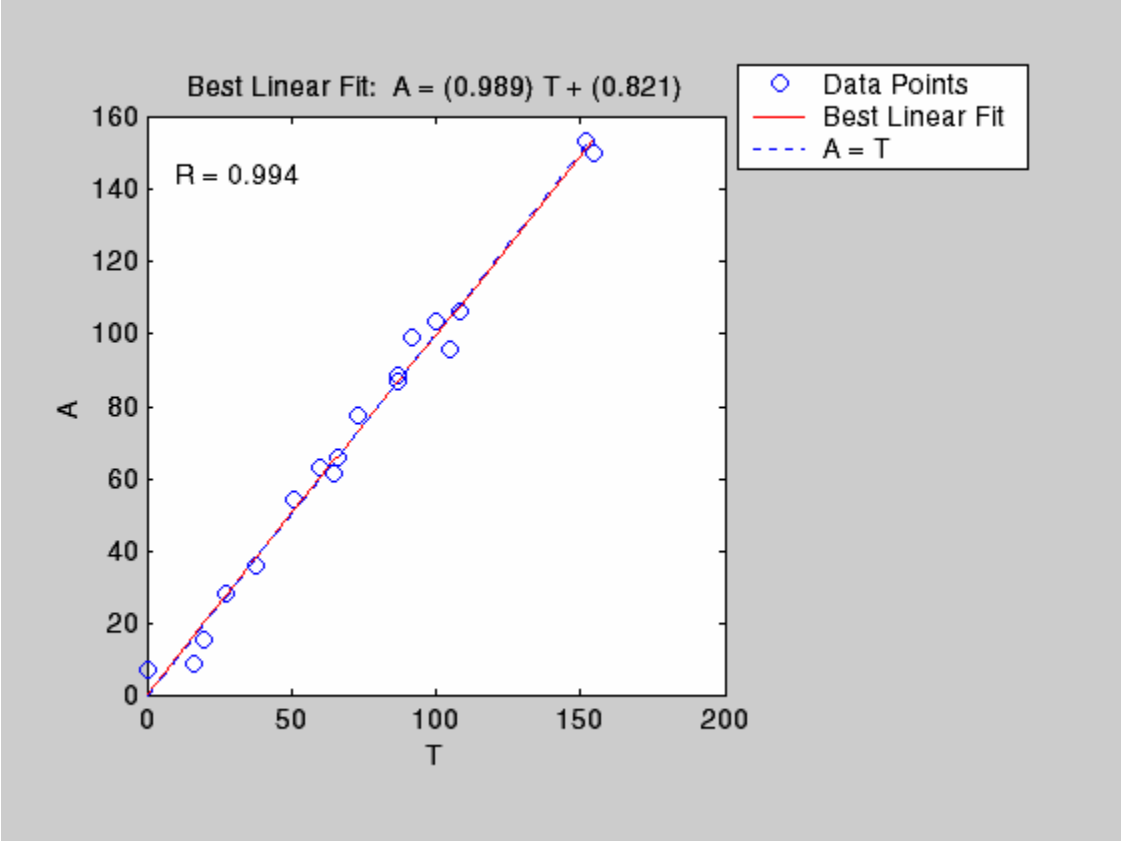


Figure 3.3 The linear regression analysis between the DMA outputs (A) for the training compounds and the corresponding actual dehalogenation rates (T).

Similarly, the QSBR model was applied to predict the dehalogenation rates of 6 testing compounds, as shown in the pipeline D (Figure 3.1). The result viewing component in pipeline D applied linear regression analysis to compare the output from DMA for 6 testing compounds with the actual dehalogenation rates, as shown in Figure 3.4. The best linear fit between the outputs and actual values is described by the equation A = 1.03T + 5.57. The slope of the linear fit, 1.03, is very close to one. The intercept, 5.57, is slightly greater than zero, which causes the best linear fit to shift

upwards to a small extent from the perfect linear fit A=T. The correlation coefficient, 0.99, indicates that 99% of the variation in the DMA outputs is explained by the actual values for the testing compounds.

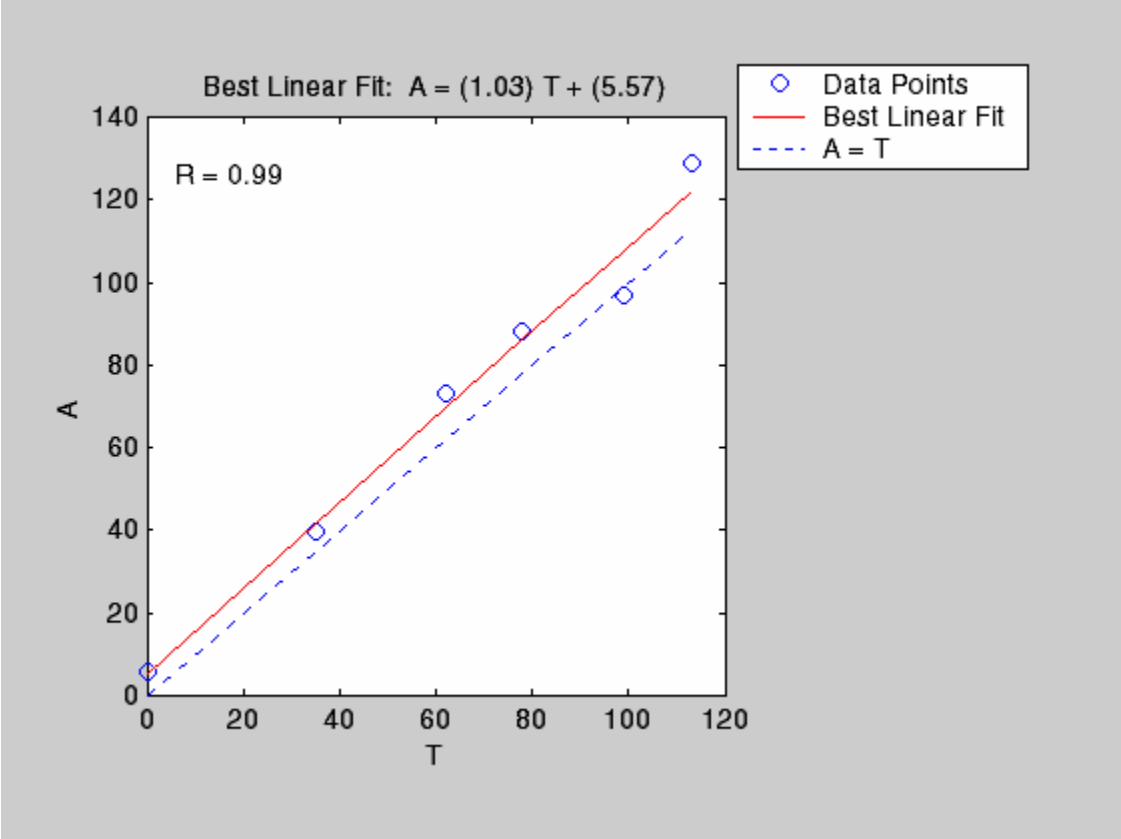


Figure 3.4. The linear regression analysis between the DMA outputs (A) for 6 testing compounds and the actual dehalogenation rates of these compounds (T).

The results from Figure 3.3 and 3.4 indicate that the data modeling component trained in the pipeline B has a relatively accurate predictive rate with the low training error.

Due to the architecture of the neural network used to implement the data modeling component and the mechanism of how it generates outputs, it is very difficult to determine which principal component has the most influence on the dehalogenation

rate. It becomes even more difficult or impossible to assess which original structural property is the primary descriptor affecting the dehalogenation rate since each principal component is a combination of nine structural properties. A neural network works like a black box. A user provides it an input, the black box generates an output. But the user can't know what actually happens in the black box, or, how it generates the output.

II. Applying Data Pipelining in Bioinformatics to Perform DNA Sequence Clustering

As of mid-2000, GeneBank contained just under 1.9 million human EST (expressed sequence tag) records [4]. The number of human genes is estimated to be 30,000 – 40,000. Without doing any sequence comparison, it is clear that each of these ESTs cannot represent a unique gene. The UniGene resource clusters ESTs and other mRNA sequences, along with coding sequences (CDSs) annotated on genomic DNA, into subsets of related sequences [5]. In most cases, sequences in each cluster are produced by a single gene, including alternatively spliced transcripts. However, some genes may be represented by more than one cluster [4].

As of July 2000, there are 1.7 million sequences belonging to 82,000 clusters in the human subset of UniGene [4]. In this section, application of data pipelining to perform DNA sequence clustering will be discussed.

A. Identifying a Core Sequence of a Cluster and Construct the 95% Confidence Interval

Eleven human clusters were downloaded from the UniGene [6]. Typically, a cluster consists of thousands of sequences that include 5'-end ESTs, 3'-end ESTs and

mRNA. For example, the cluster Hs.1516 (cluster identifier where Hs represents *Homo sapiens*) located in the 17th human chromosome contains 1123 sequences. A C++ program was written to select the 1st 100 5'-end ESTs from each cluster for the purpose of sample analysis. The C++ program also picked the next 10 5'-end ESTs for the testing purpose.

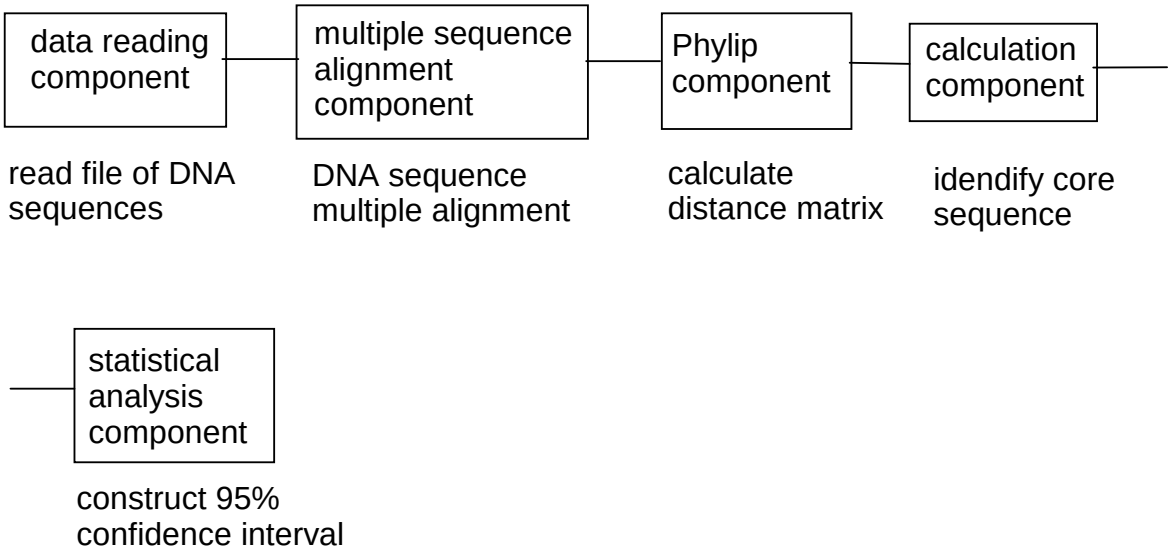


Figure 3.5 The data pipeline used to identify a core sequence to represent a cluster and construct 95% confidence distance interval for that cluster.

Figure 3.5 demonstrates the data pipeline used to cluster ESTs. To better illustrate the data flow over the data pipeline, a sample data set, which contains ten 5'-end ESTs belonging to the cluster Hs.1516, is used. The sample data set is shown below:

```
> 459146 gnl|UG|Hs#S459146 zk34a07.r1 Homo sapiens cDNA, 5' end
/clone=IMAGE:484692 /clone_end=5' /gb=AA037567 /gi=1512667 /ug=Hs.1516
/len=468
ATCGTCCTTCCTCTCAAGCTAGCCAGAGGGTGGGAGCCTAAGGAAGCGTGGGGTAGCAGA
TGGAGTAATGGTCACGAGGTCCAGACCCACTCCCAAAGCTCAGACTTGCCAGGCTCCCTT
TCTCTTCTTCCCCAGGTCCTTCCTTTAGGTCTGGTTGTTGCACCATCTGCTTGGTTGGCT
GGCAGCTGAGAGCCCTGCTGTGGGAGAGCGAAGGGGGTCAAAGGAAGACTTGAAGCACAG
AGGGCTAGGGAAGGTGGGGTACATTTCTCTGAAGCAGTCAGGGTGGGAAGAAAGAATGCA
```

AGAAGTGGACTTGAATGTGCCCTAATGGAGAAGACCCCACCGTTGCTANGGGGAATGGAG
GGGCTTTCCTGGGGNNCCTGGTTCCCCTAACCCCATTTTNGTGGGTCCACAAGCCATGAA
AGTCACCGGGAATGAACCTATCCTTCCAGTGGCTCGCTCCCTGTAGCT
> 27025 gnl|UG|Hs#S27025 EST88091 Homo sapiens cDNA, 5' end
/clone=ATCC:106939 /clone_end=5' /gb=T29629 /gi=611727 /ug=Hs.1516
/len=396
GGACATTTTTTGGTTTTNTNCTGTTTTGTTAAAAAAAAGAAAAAGAAGAAAAGACATCAT
GGCCAACTGGTAGGTTCCTAAGTNTCCTTCCATCCAGTCAAGCCAGAAGATGCCCAGGGG
AGTAGGGAGGTNTNGGGAGGGAGGTGTAGGGGAAGGAGATATGGAGAGGGAGGCAGAGCT
ACAGGGAGCGAGCCACTGGAAGGATAGGTTCATCCCGGTGACTTCATGGCTGTNACCACA
AATGGGGTAGGGAACAGGACCCAGGAAGCCCCTCATCCCCTAGCACGTGGGTCTTCTCCA
TTAGGCACATTTCAGTCCACTTTTTGCATTCTTTTCTTNCCAACCCTGACTTGTTCAGAG
GAATGTTACCCCACCTNCCTTAGCCCTTTGTGCTTA
> 302512 gnl|UG|Hs#S302512 yu82g04.r1 Homo sapiens cDNA, 5' end
/clone=IMAGE:240342 /clone_end=5' /gb=H89809 /gi=1080239 /ug=Hs.1516
/len=394
ATTCGGCACAGGAGACATGTACCTTGACCATCGTCCTTCCTCTCAAGCTAGCCAGAGGGT
GGGAGCCTAAGGAAGCGTGGGGTAGCAGATGGAGTAATGGTCACGAGGTCCAGACCCACT
CCCAAAGCTCAGACTTGAAGCACAGAGGGCTAGGGAGGTGGGGTACATTTCTCTGAGCAG
TCAGGGTGGGAAGAAAGAATGCAAGAGTGGACTGAATGTGCCTAATGGAGAAGACCCACG
TGCTAGGGGATGAGGGGCTTCCTGGGGTCCTGTTCCCTACCCCATTTGTGGTCACAGCCA
TGAAGTCACCGGGATGANCCTATCCTTCCAGTTGGCTCGCTCCCTGTAGCTCTGCCTNCC
TTCTCCATAATCTTCCTTTCCCTAACAACTTCCT
> 2282903 gnl|UG|Hs#S2282903 601473253F1 Homo sapiens cDNA, 5' end
/clone=IMAGE:3876205 /clone_end=5' /gb=BE619428 /gi=9890366 /ug=Hs.1516
/len=155
ATCTCCTTCCCCTACACCTCCCTCCCCACACCTCCCTACTCCCCTGGGCATCTTCTGGCT
TGACTGGATGGAAGGAGACTTAGGAACCTACCAGTTGGCCATGATGTCTTTTCTTCTTTT
TCTTTTTTTTAACAAAACAGAACAAAACCAAAAAA
> 2369565 gnl|UG|Hs#S2369565 601474188F1 Homo sapiens cDNA, 5' end
/clone=IMAGE:3877124 /clone_end=5' /gb=BE784560 /gi=10205845
/ug=Hs.1516 /len=151
TCCTTCCCCTACACCTCCCTCCCCACACCTCCCTACTCCCCTGGGCATCTTCTGGCTTGA
CTGGATGGAAGGAGACTTAGGAACCTACCAGTGGCCATGATGTCTTTTCTTCTTTTTCTT
TTTTTTAACAAAACAGAACAAAACCAAAAAA
> 2474669 gnl|UG|Hs#S2474669 601682342F1 Homo sapiens cDNA, 5' end
/clone=IMAGE:3952451 /clone_end=5' /gb=BE898918 /gi=10365882
/ug=Hs.1516 /len=620
CTCAGACTTGCCAGGCTCCCTTTCTCTTCTTCCCCAGGTCCTTCCTTTAGGTCTGGTTGT
TGCACCATCTGCTTGGTTGGCTGGCAGCTGAGAGCCCTGCTGTGGGAGAGCGAAGGGGGT
CAAAGGAAGACTTGAAGCACAGAGGGCTAGGGAGGTGGGGTACATTTCTCTGAGCAGTCA
GGGTGGGAAGAAAGAATGCAAGAGTGGACTGAATGTGCCTAATGGAGAAGACCCACGTGC
TAGGGGATGAGGGGCTTCCTGGGTCCTGTTCCCTACCCCATTTGTGGTCACAGCCATGAA
GTCACCGGGATGAACCTATCCTTCCAGTGGCTCGCTCCTGTAGCTCTGCTCCCTCTCCAT
ATTTTCTTTCCCCTAAAACTCCTCCCCAAAACTCCCTAATCCCCTGGGCATCTTCTGGTT
GACTGTTTGGGAGGGACTTAGGAACTACAGGTGGGCATGATGTCTTTCTTCTTTTCCTTT
TTTTACCACAACAGACCAAACCAATATGTCCGAAAAAAAAAAAAAAAATCGCGGCTCTTTC
GGGGGCGCACAAGGGTGGAAGGGCCGGGCTACTTGTCCGCCCCTTTGTTCAAGGGAAAGC
CGGCCAAAGAGCGAAATGAC
> 648248 gnl|UG|Hs#S648248 EST32906 Homo sapiens cDNA, 5' end
/clone=ATCC:130960 /clone_end=5' /gb=AA329312 /gi=1981556 /ug=Hs.1516
/len=178
GCTCCCTGTAGCTCTGCCTCCCTCTCCATATCTCCTTCCCCTACACCTCCCTCCCCACAC
CTCCCTACTCCCCTGGGCATCTTCTGGCTTGACTGGATGGAAGGAGACTTAGGAACCTAC
CAGTTGGCCATGATGTCTTTTCTTCTTTTTCTTTNTTTTAACAAAACAGAACAAAACC

```
> 2282904 gnl|UG|Hs#S2282904 601473255F1 Homo sapiens cDNA, 5' end
/clone=IMAGE:3876301 /clone_end=5' /gb=BE619429 /gi=9890367 /ug=Hs.1516
/len=787
GAAGGGGGTCAAAGGAAGACTTGAAGCACAGAGGGCTAGGGAGGTGGGGTACATTTCTCT
GAGCAGTCAGGGTGGGAAGAAAGAATGCAAGAGTGGACTGAATGTGCCTAATGGAGAAGA
CCCACGTGCTAGGGGATGAGGGGCTTCCTGGGTCCTGTTCCCTACCCCATTTGTGGTCAC
AGCCATGAAGTCACCGGGATGAACCTATCCTTCCAGTGGCTCGCTCCCTGTAGCTCTGCC
TCCCTCTCCATATCTCCTTCCCCTACACCTCCCTCCCCACACCTCCCTACTCCCCTGGGC
ATCTTCTGGCTTGACTGGATGGAAGGAGACTTAGGAACCTACCAGTTGGCCATGATGTCT
TTTCTTCTTTTTCTTTTTTTTAACAAAACAGAACAAAACCAAAAAATGTCCAGAAAAAAA
CAACACAAAAAAAAAAAAACACACAGAGCAAAAGAGAAACACAAAAGCAACCGACCCGCG
CCGCGCAGCGAAGAAAAAAGGAGAGAAGAAGAGGGGAGCAGACAGAGGAGAAGAAGCAAG
GAGAGGCAAGCGGAGCGACACGATGGAAGAGGGCGAAGAGAAGAGAAGAAGAAAGAACAA
GAAGACAGAACGAGTAGGTGGGCAGTAGGGTGTAGGAGGAAAGGGAGGGAGGGAGAGAGG
AGCAAAGGAGAGAGCAAGGGAGGGAGAGAAGGAGAGCGAGAGGGAAGGGGGAGAGGAGAG
AGGAGGGCGAAGGCGGTTCTGTTGTGAGGAGGAGGGCGAGGAGGAGGGGGGAGAGGGCAG
ACCGCGG
> 1597887 gnl|UG|Hs#S1597887 df29g03.y1 Homo sapiens cDNA, 5' end
/clone=IMAGE:2485036 /clone_end=5' /gb=AW021895 /gi=5875425 /ug=Hs.1516
/len=350
GCACGAGGTGGACTGAATGTGCCTAATGGAGAAGACCCACGTGCTAGGGGATGAGGGGCT
TCCTGGGTCCTGTTCCCTACCCCATTTGTGGTCACAGCCATGAAGTCACCGGGATGAACC
TATCCTTCCAGTGGCTCGCTCCCTGTAGCTCTGCCTCCCTCTCCATATCTCCTTCCCCTA
CACCTCCCTCCCCACACCTCCCTACTCCCCTGGGCATCTTCTGGCTTGACTGGATGGAAG
GAGACTTAGGAACCTACCAGTTGGCCATGATGTCTTTTCTTCTTTTTCTTTTTTTTAACA
AAACAGAACAAAACCAAAAAATGTCCAGAAAAAAAAAAAAAAAAAAAAA
> 3895887 gnl|UG|Hs#S3895887 603047615F1 Homo sapiens cDNA, 5' end
/clone=IMAGE:5188011 /clone_end=5' /gb=BI763259 /gi=15754837
/ug=Hs.1516 /len=466
CAGCTGAGAGCCCTGCTGTGGGAGAGCGAAGGGGGTCAAAGGAAGACTTGAAGCACAGAG
GGCTAGGGAGGTGGGGTACATTTCTCTGAGCAGTCAGGGTGGGAAGAAAGAATGCAAGAG
TGGACTGAATGTGCCTAATGGAGAAGACCCACGTGCTAGGGGATGAGGGGCTTCCTGGGT
CCTGTTCCCTACCCCATTTGTGGTCACAGCCATGAAGTCACCGGGATGAACCTATCCTTC
CAGTGGCTCGCTCCCTGTAGCTCTGCCTCCCTCTCCATATCTCCTTCCCCTACACCTCCC
TCCCCACACCTCCCTACTCCCCTGGGCATCTTCTGGCTTGACTGGATGGAAGGAGACTTA
GGAACCTACCAGTTGGCCATGATGTCTTTTCTTCTTTTTCTTTTTTTTAACAAAACAGAA
CAAAACCAAAAAATGTCCAAAAAAAAAAACAAAAAAAAAAAAAGGG
```

The data pipeline first read the file of the sample data set. The multiple sequence alignment component applied the Clustal W to align the ten sequences [7]. In the configuration window of the multiple sequence alignment component, the output was set to be the Phylip format. The following shows the multiple alignment output of ten ESTs in Phylip format.

```
    10    901
459146     ATCGTCCTTC CTCTCAAGCT AGCCAGAGGG TGGGAGCCTA AGGAAGCGTG
302512     ---------- ---------- ---------- ---------- ----------
648248     ---------- ---------- ---------- ---------- ----------
2282904    ---------- ---------- ---------- ---------- ----------
```

```
2282903    ---------- ---------- ---------- ---------- ----------
2369565    ---------- ---------- ---------- ---------- ----------
1597887    ---------- ---------- ---------- ---------- ----------
3895887    ---------- ---------- ---------- ---------- ----------
2474669    CTCAGACTTG CCAGGCTCCC TTTCTCTTCT TCCCCAGGTC CTTCCTTTAG
27025      ---------- ---------- ---------- ---------- ----------

           GGGTAGCAGA TGGAGTAATG GTCACGAGGT CCAGACCCAC TCCCAAAGCT
           ---------- ---------- ---------- ---------- --------AT
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ----CAGCTG AGAGCCCTGC
           GTCTGGTTGT TGCACCATCT GCTTGGTTGG CTGGCAGCTG AGAGCCCTGC
           ---------- ---------- ---------- ---------- ----------

           CAGACTTGCC AGGCTCCCTT TCTCTTCTTC CCCAGGTCCT TCCTTTAGGT
           TCGGCACAGG AGACATGTAC CTTGACCATC GTCCTTCCTC TCAAGCTAGC
           ---------- ---------- ---------- ---------- ----------
           ---------- -GAAGGGGGT CAAAGGAAGA CTTGAAGCAC AGAGGGCTAG
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           TGTGGGAGAG CGAAGGGGGT CAAAGGAAGA CTTGAAGCAC AGAGGGCTAG
           TGTGGGAGAG CGAAGGGGGT CAAAGGAAGA CTTGAAGCAC AGAGGGCTAG
           ------GGAC ATTTTTTGGT TTTNTNCTGT TTTGTTAAAA AAAAGAAAAA

           CTGGTTGTTG CACCATCTGC TTG--GTTGG CTGGCAGCTG AGAGCCCTGC
           CAGAGGGTGG GAGCCTAAGG AAG-CGTGGG GTAGCAGATG GAGTAATGGT
           ---------- ---------- ---------- ---------- ----------
           GGAGGTGGGG TACATTTCTC TGA-GCAGTC AGGGTGGGAA GAAAGAATGC
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- -------GCA
           GGAGGTGGGG TACATTTCTC TGA-GCAGTC AGGGTGGGAA GAAAGAATGC
           GGAGGTGGGG TACATTTCTC TGA-GCAGTC AGGGTGGGAA GAAAGAATGC
           GAAGAAAAGA CATCATGGCC AACTGGTAGG TTCCTAAGTN TCCTTCCATC

           TGTGGGAGAG CGAAGGGGGT CAAAGGA--A GACTTGAAGC ACAGAGGGCT
           CACGAGGTCC AGACCCACTC CCAAAGCTCA GACTTGAAGC ACAGAGGGCT
           ---------- ---------- ---------- ---------- ----------
           AAGAGTGGAC TGAATGTGCC TAATGGAGAA GACCCACGTG CTAGGGGATG
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
           CGAGGTGGAC TGAATGTGCC TAATGGAGAA GACCCACGTG CTAGGGGATG
           AAGAGTGGAC TGAATGTGCC TAATGGAGAA GACCCACGTG CTAGGGGATG
           AAGAGTGGAC TGAATGTGCC TAATGGAGAA GACCCACGTG CTAGGGGATG
           CAGTCAAGCC AGAAGATGCC CAGGGGAGTA GG---GAGGT NTNGGGAGGG

           AGGGAA-GGT GGGGTACATT TCTCTGAAGC AGTCAGGGTG GGAAGAAAGA
           AGGGA--GGT GGGGTACATT TCTCTGA-GC AGTCAGGGTG GGAAGAAAGA
           ---------- ---------- ---------- ---------- ----------
           AGGGGCTTCC TGGGTCCTGT TCCCTACCCC ATTTGTGGTC ACAGCCATGA
           ---------- ---------- ---------- ---------- ----------
           ---------- ---------- ---------- ---------- ----------
```

```
AGGGGCTTCC TGGGTCCTGT TCCCTACCCC ATTTGTGGTC ACAGCCATGA
AGGGGCTTCC TGGGTCCTGT TCCCTACCCC ATTTGTGGTC ACAGCCATGA
AGGGGCTTCC TGGGTCCTGT TCCCTACCCC ATTTGTGGTC ACAGCCATGA
AGGTGTAGGG GAAGGAGATA TGGA-GAGGG AGGCAGAGCT ACAGGGAGCG

ATGCAAGAAG TGGACTTGAA TGTGCCCTAA TGGAGAAGAC CCCACCGTTG
ATGCAAGA-G TGGACT-GAA TGTGCC-TAA TGGAGAAGAC CC--ACGT-G
---------- ---------- ---------- ----GCTCCC TGTAGCTCTG
AGTCACCGGG ATGAACCTAT CCTTCCAGTG GCTCGCTCCC TGTAGCTCTG
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
AGTCACCGGG ATGAACCTAT CCTTCCAGTG GCTCGCTCCC TGTAGCTCTG
AGTCACCGGG ATGAACCTAT CCTTCCAGTG GCTCGCTCCC TGTAGCTCTG
AGTCACCGGG ATGAACCTAT CCTTCCAGTG GCTCGCTCC- TGTAGCTCTG
AGCCACTGGA AGGATAGGTT CATCCCGGTG ACTTCATGGC TGTNACCACA

CTANGGGGAA TGGAGGGGCT TT-CCTGGGG NNCCTGGTTC CCCTAACCCC
CTAGGGGAT- --GAGGGGCT T--CCTGGGG TCCT--GTTC CCT--ACCCC
CCTCCCTCTC CATATCTCCT TCCCCTACAC CTCCCTCCCC ACACCTCCCT
CCTCCCTCTC CATATCTCCT TCCCCTACAC CTCCCTCCCC ACACCTCCCT
---------- ---ATCTCCT TCCCCTACAC CTCCCTCCCC ACACCTCCCT
---------- ------TCCT TCCCCTACAC CTCCCTCCCC ACACCTCCCT
CCTCCCTCTC CATATCTCCT TCCCCTACAC CTCCCTCCCC ACACCTCCCT
CCTCCCTCTC CATATCTCCT TCCCCTACAC CTCCCTCCCC ACACCTCCCT
C-TCCCTCTC CATATTTTCT TTCCCCTAAA ACTCCTCCCC AAAACTCCCT
A-ATGGGGTA GGGAACAGGA CCCAGGAAGC CCCTCATCCC CTAGCACGTG

ATTTTNGTGG GT-CCACAAG CCATGAAAGT CACCGGGAAT GAACCTAT--
ATT--TGTGG ---TCACA-G CCATGAA-GT CACCGGGA-T GANCCTAT--
ACTCCCCTGG GCATCTTCTG GCTTGAC--T GGATGGAAGG AGACTTAGGA
ACTCCCCTGG GCATCTTCTG GCTTGAC--T GGATGGAAGG AGACTTAGGA
ACTCCCCTGG GCATCTTCTG GCTTGAC--T GGATGGAAGG AGACTTAGGA
ACTCCCCTGG GCATCTTCTG GCTTGAC--T GGATGGAAGG AGACTTAGGA
ACTCCCCTGG GCATCTTCTG GCTTGAC--T GGATGGAAGG AGACTTAGGA
ACTCCCCTGG GCATCTTCTG GCTTGAC--T GGATGGAAGG AGACTTAGGA
AATCCCCTGG GCATCTTCTG G-TTGAC--T GTTTGGGAGG -GACTTAGGA
GGTCTTCT-- ---CCATTAG GCACATTTCA GTCCACTTTT TGCATTCTTT

-CCTTCCAGT -GGCT----C GCTCCCTGTA GCT------- ----------
-CCTTCCAGT TGGCT----C GCTCCCTGTA GCTCTGCCTN CCTTCTCCAT
ACCTACCAGT TGGCCATGAT GTCTTTTCTT CTTTTTCTTT NTTTTAACAA
ACCTACCAGT TGGCCATGAT GTCTTTTCTT CTTTTTCTTT TTTTTAACAA
ACCTACCAGT TGGCCATGAT GTCTTTTCTT CTTTTTCTTT TTTTTAACAA
ACCTACCAGT -GGCCATGAT GTCTTTTCTT CTTTTTCTTT TTTTTAACAA
ACCTACCAGT TGGCCATGAT GTCTTTTCTT CTTTTTCTTT TTTTTAACAA
ACCTACCAGT TGGCCATGAT GTCTTTTCTT CTTTTTCTTT TTTTTAACAA
AC-TACAGGT -GGGCATGAT GTCTTTCTT- -CTTTTCCTT TTTTTACCAC
TCTTNCCAAC --CCTGACTT GTTCAGAGGA ATGTTACCCC ACCTN-CCTT

---------- ---------- ---------- ---------- ----------
AATCTTCCTT TCCCTAACAA CTTCCT---- ---------- ----------
AACAGAACAA AACC------ ---------- ---------- ----------
AACAGAACAA AACCAAAAAA TGTCCAGAAA AAAACAACAC AAAAAAAAAA
AACAGAACAA AACCAAAAAA ---------- ---------- ----------
AACAGAACAA AACCAAAAAA ---------- ---------- ----------
AACAGAACAA AACCAAAAAA TGTCCAGAAA AAAAAAAAAA AAAAAAAAA-
AACAGAACAA AACCAAAAAA TGTCCAAAAA AAAAAACAAA AAAAAAAAAA
```

```
AACAGACCAA A-CCAATATG TCCGAAAAAA AAAAAAAAAT CGCGGCTCTT
AGCCCTTTGT GCTTA----- ---------- ---------- ----------

---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
AAACACACAG AGCAAAAGAG AAACACAAAA GCAACCGACC CGCGCCGCGC
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
GGG------- ---------- ---------- ---------- ----------
TCGGGGGCGC ACAAGGGTGG AAGGGCCGGG CTACTTGTCC GCCCCTTTGT
---------- ---------- ---------- ---------- ----------

---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
AGCGAAGAAA AAAGGAGAGA AGAAGAGGGG AGCAGACAGA GGAGAAGAAG
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
TCAAGGGAAA GCCGGCCAAA GAGCGAAATG AC-------- ----------
---------- ---------- ---------- ---------- ----------

---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
CAAGGAGAGG CAAGCGGAGC GACACGATGG AAGAGGGCGA AGAGAAGAGA
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------

---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
AGAAGAAAGA ACAAGAAGAC AGAACGAGTA GGTGGGCAGT AGGGTGTAGG
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------

---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
AGGAAAGGGA GGGAGGGAGA GAGGAGCAAA GGAGAGAGCA AGGGAGGGAG
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
```

```
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
AGAAGGAGAG CGAGAGGGAA GGGGGAGAGG AGAGAGGAGG GCGAAGGCGG
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------

---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
TTCTGTTGTG AGGAGGAGGG CGAGGAGGAG GGGGGAGAGG GCAGACCGCG
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------

-
-
-
G
-
-
-
-
-
-
```

It can be seen that the original ten sequences vary in length, but the resulting aligned sequences in the Phylip format have the same length (901 bp) by adding the heading and trailing consecutive gaps.

The Phylip component calculated a distance between any two sequences among the ten multiple aligned sequences. The distance is an inverse measurement of the similarity between two sequences. The distance between a sequence and itself is zero. The result of the Phylip component, a distance matrix, is shown below:

	459146	302512	648248	2282904	2282903	2369565	1597887	3895887	2474669	27025
459146	0.0000	0.3395	1.3490	1.9841	1.0431	1.0096	1.4486	2.4239	3.2881	1.8964
302512	0.3395	0.0000	1.3055	1.8778	1.0532	1.0418	1.5116	1.9057	2.1233	1.6695
648248	1.3490	1.3055	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1797	2.9492
2282904	1.9841	1.8778	0.0000	0.0000	0.0000	0.0000	0.0294	0.0185	0.2475	2.5662
2282903	1.0431	1.0532	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2386	2.6062
2369565	1.0096	1.0418	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2351	2.6762
1597887	1.4486	1.5116	0.0000	0.0294	0.0000	0.0000	0.0000	0.0263	0.1737	2.1128
3895887	2.4239	1.9057	0.0000	0.0185	0.0000	0.0000	0.0263	0.0000	0.1140	2.5565
2474669	3.2881	2.1233	0.1797	0.2475	0.2386	0.2351	0.1737	0.1140	0.0000	2.4989
27025	1.8964	1.6695	2.9492	2.5662	2.6062	2.6762	2.1128	2.5565	2.4989	0.0000

The distance matrix has 11 rows and 11 columns. The 1st column and 1st row represent the sequence numbers of ten sequences. Ten values in the 2nd row, 0.0000, 0.3395, 1.3490, …, 1.8964 represent the distance between the sequence 459146 and itself, the distance between the sequence 459146 and the sequence 302512, the distance between the sequence 459146 and the sequence 648248, ..., the distance between the sequence 459146 and the sequence 27025, respectively. The left rows represent distance values analogously. Occasionally, if the distance between two sequences is too large or the similarity is too small, the distance value becomes NaN (not a number).

The calculation component selects one sequence from ten ESTs and uses it as a core sequence to represent them based on the distance matrix. For each of the ten sequences, the component calculates its mean square distance (MSD), which is defined in equation 3.1.

$$MSD(i) = (d_{1,i}^2 + d_{2,i}^2 + \ldots + d_{10,i}^2) / 10 \qquad [3.1]$$

In this equation, i represents the i^{th} sequence in the distance matrix, and $d_{n,i}$ represents the distance between the n^{th} sequence and i^{th} sequence. The sequence with the smallest MSD among the ten sequences is chosen as the core sequence. In our example, the 7th

sequence, sequence 1597887, is determined to be the core sequence to represent the ten sequences since it has the minimal MSD, 0.8879.

Among the ten distance values to the core sequence, any distance greater than 1 is considered to be an outlier and is removed by the statistical analysis component. In our example, three sequences whose distances to the core sequence 1597887 are greater than 1 are removed as outliers. The component then makes the histogram to show the distribution of the remaining seven distance values to the core sequence (Figure 3.6). In the histogram, the x-axis reflects the range of distances to the core sequence, 0 – 0.1737, which is divided into 20 equally spaced containers. The y-axis shows the number of distance values that fall within the containers.

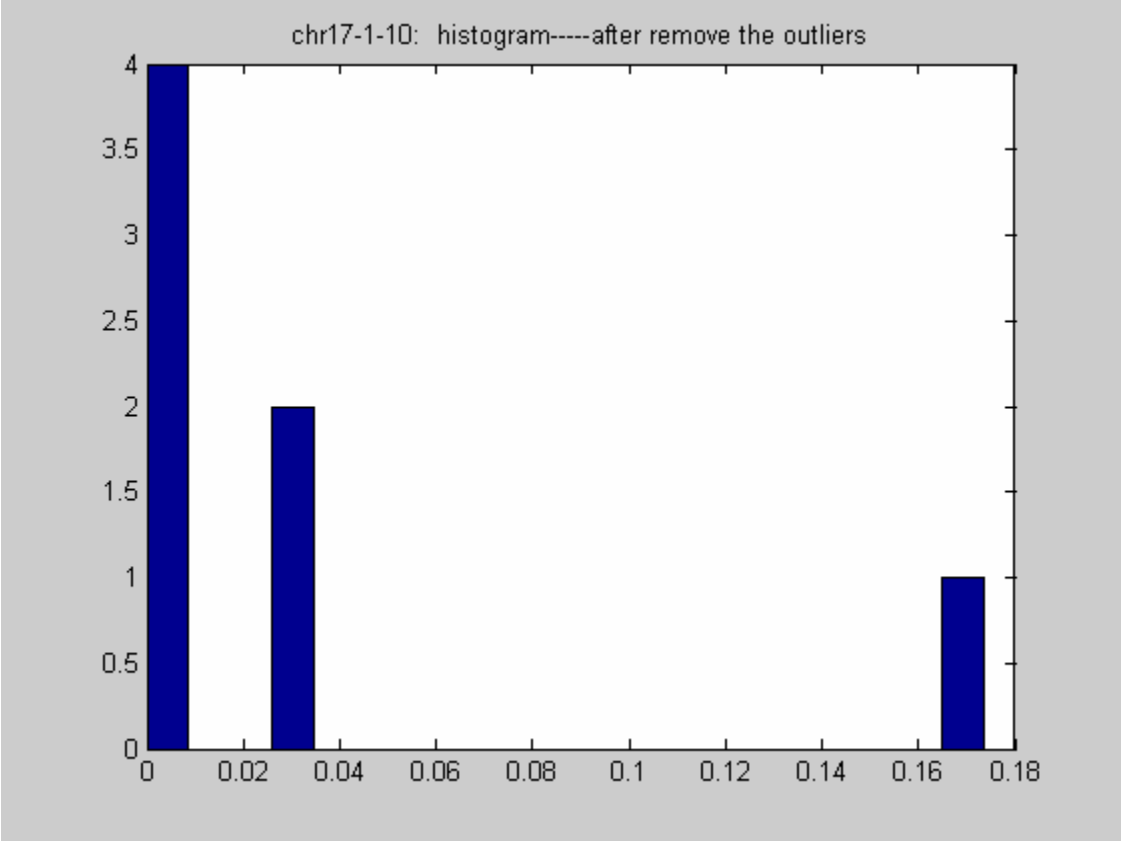


Figure 3.6 The histogram of distances to the core sequence 1597887 after removing outliers.

Assuming the distribution of the distances to the core sequence 1597887 is a gamma distribution, the statistical analysis component calculates the maximum likelihood estimates (MLEs) for the parameters of the gamma distribution. Then it constructs the 95% confidence interval of the gamma distribution, which is from 0 to 0.2044.

In the real study of DNA sequence clustering, 100 5'-end ESTs of a cluster were used to identify the core sequence and construct the 95% confidence distance interval for that cluster. In the computation of MSDs, equation 3.1 is used, but only $d_{n,i}$ whose value is a real number and is less than 10 is considered. Figure 3.7 shows the histogram of distribution of distances to the core sequence of the cluster Hs.78771 located in the x chromosome after removing outliers. From the shape of the histogram, the distribution was estimated and tested to be the gamma distribution. Its 95% confidence distance interval was from 0 to 0.1583.

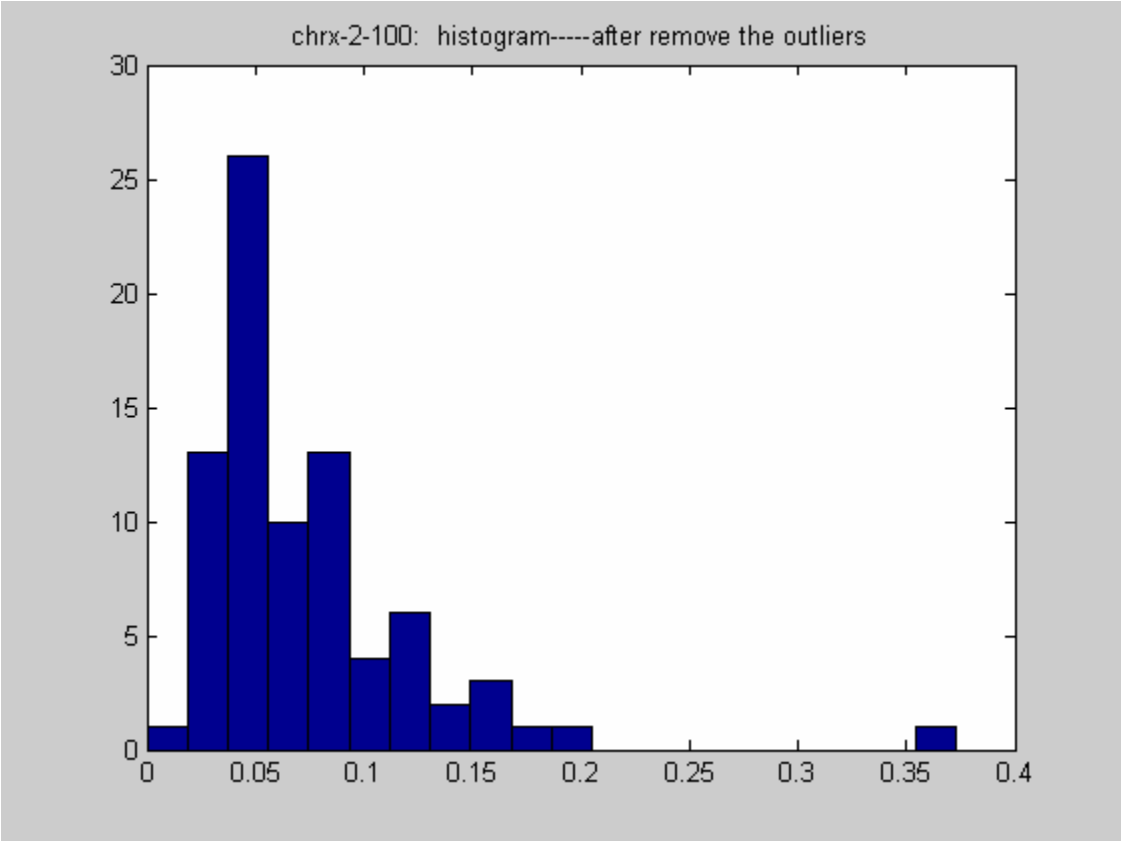


Figure 3.7 The histogram of distribution of distances to the core sequence of the cluster Hs.78771 after removing outliers.

The core sequence number, the 95% confidence distance interval, and the number of sequences whose distances to the core sequence are less than 1 among the 100 5'-end ESTs for each of the 11 clusters are summarized in Table 3.3.

Table 3.3 Eleven clusters and their core sequences, 95% confidence distance intervals

Chromosome	Cluster	Core sequence number	95% confidence distance interval		Number of sequences*
17	Hs.1516	43503	0.0000	0.4386	50
17	Hs.79474	676622	0.0000	0.2471	57
18	Hs.155101	2428408	0.0000	0.2117	62
19	Hs.343354	2351216	0.0000	0.0179	75
19	Hs.111334	252674	0.0000	0.2143	59
20	Hs.179666	656436	0.0000	0.0890	54
20	Hs.194676	2429741	0.0000	0.5853	81
21	Hs.85119	698464	0.0000	0.5372	33
22	Hs.181125	2218749	0.0000	0.1008	100
X	Hs.78771	597966	0.0000	0.1583	81
Y	Hs.180911	658694	0.0000	0.1291	79

* The number of sequences whose distances to the core sequence is less than 1.0 among the 100 selected ESTs.

B. Clustering a New Sequence

For a new EST, one might ask: to which cluster does it belong? The following pipeline, as shown in Figure 3.8, serves to cluster a new sequence.

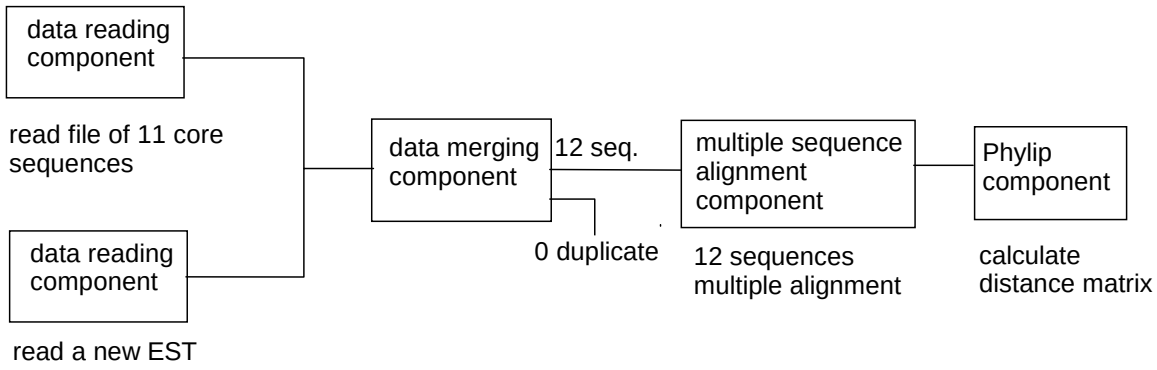


Figure 3.8 A data pipeline functioning to cluster a new sequence.

In the pipeline, the data reading component read the file that contains 11 core sequences representing 11 clusters and read a 5'-end new EST. These 11 core sequences and new EST were merged by the data merging component. The multiple sequence alignment component and Phylip component did the same thing for the 12 sequences as they did for the sample data set. In the resulting distance matrix, the distance between the new EST and each of 11 core sequences was examined. If the distance between the new EST and one core sequence representing the cluster i is within the 95% confidence distance interval of the cluster i, then the new EST belongs to that cluster. If the distance is larger than the 95% confidence interval, then no conclusion can be made.

As mentioned earlier, the C++ program selected ten 5'-end ESTs for the testing purpose for each of the 11 clusters. Each of the ten testing ESTs was merged with the 11 core sequences, and its distance to each core sequence was examined. For example, the sequence 2186756 is one of ten testing ESTs belonging to the cluster Hs.78771. It was merged with the 11 core sequences and the distance matrix is shown in the following:

	597966	2186756	656436	2428408	2351216	658694	698464	2429741	676622	43503	2218749	252674
597966	0.0000	0.0375	1.0362	1.5393	1.1724	1.8829	3.3610	2.6676	2.3165	2.9738	3.0390	2.6107
2186756	0.0375	0.0000	1.0352	1.6662	1.2108	1.9603	3.7646	2.9183	2.2193	3.2450	3.7705	2.6782
656436	1.0362	1.0352	0.0000	1.1588	1.4928	1.8890	2.5153	2.8292	2.2908	2.9360	2.2131	3.4095
2428408	1.5393	1.6662	1.1588	0.0000	1.9322	2.0726	3.0319	2.6463	4.2610	2.6295	2.8036	3.6122
2351216	1.1724	1.2108	1.4928	1.9322	0.0000	1.8951	2.3965	2.3776	2.2809	2.5424	2.3507	2.2431
658694	1.8829	1.9603	1.8890	2.0726	1.8951	0.0000	2.0137	3.3446	2.4148	6.0600	2.2845	2.5615
698464	3.3610	3.7646	2.5153	3.0319	2.3965	2.0137	0.0000	6.0600	5.4619	3.0001	3.1561	3.4809
2429741	2.6676	2.9183	2.8292	2.6463	2.3776	3.3446	6.0600	0.0000	6.0600	6.0600	4.4269	6.0600
676622	2.3165	2.2193	2.2908	4.2610	2.2809	2.4148	5.4619	6.0600	0.0000	2.9281	3.7138	4.7808
43503	2.9738	3.2450	2.9360	2.6295	2.5424	6.0600	3.0001	6.0600	2.9281	0.0000	2.9214	2.9701
2218749	3.0390	3.7705	2.2131	2.8036	2.3507	2.2845	3.1561	4.4269	3.7138	2.9214	0.0000	2.0725
252674	2.6107	2.6782	3.4095	3.6122	2.2431	2.5615	3.4809	6.0600	4.7808	2.9701	2.0725	0.0000

The 3rd row lists the distance between the sequence 2186756 and each of the 11 core sequences. The value 0.0 in the 3rd row is the distance between the sequence 2186756 and itself. We can see that the distance between the testing sequence and the core sequence 597966 representing the cluster Hs.78771, 0.0375, is within the 95% confidence interval of the cluster Hs.78771, 0 – 0.1583 (Table 3.3). Thus, the testing sequence belongs to the cluster Hs.78771 according to our hypothesis. This is a correct clustering since the testing sequence was chosen from the cluster Hs.78771. We can also see that the distance values between the testing sequence and the other core sequences are all greater than 1, which are outside the 95% confidence intervals. Thus there is no false clustering.

Table 3.4 summarizes the testing results for all of the 110 testing ESTs selected from the 11 clusters.

Table 3.4 The testing results of clustering new sequences

Cluster	Number of testing sequences	Number of correct clustering*	Number of false clustering**
Hs.1516	10	5	0
Hs.79474	10	4	0
Hs.155101	10	10	0
Hs.343354	10	9	0
Hs.111334	10	9	0
Hs.179666	10	3	0
Hs.194676	10	10	0
Hs.85119	10	3	0
Hs.181125	10	10	0
Hs.78771	10	9	0
Hs.180911	10	9	0

*Correct clustering means that the distance between a testing EST chosen from the cluster i and the core sequence of cluster i is within that cluster's 95% confidence distance interval.
**False clustering means the distance between a testing EST of cluster i and the core sequence of cluster j ($i \neq j$) is within the 95% confidence distance interval of the cluster j.

Thus, among 110 testing ESTs, 81 (74%) were correctly clustered, and there was no false clustering, which validates our hypothesis.

References

1. Leach, A. R. *Molecular Modelling* Addison Wesley Longman Limited: Singapore, 1996.

2. Damborsky, J.; Manova, K.; Kuty, M. A Mechanistic Approach to Deriving Quantitative Structure-Biodegradability Relationships, *In: W.J.G.M. Peijnenburg and J. Damborsky (Eds.), Biodegradability Prediction*. Kluwer Academic Publishers: Dordrecht, pp. 75-92, 1996.

3. Demuth, H.; Beale, M. *Neural Network Toolbox User's Guide* Mathworks Inc., 2001.

4. *Bioinformatics: A Practical Guide to the Analysis of Genes and Protein*. Baxevanis, A. D.; Francis Ouellette, B. F., Eds.; John Wiley & Sons: Denver, 2001.

5. Boguski, M. S.; Schuler, G. D. Establishing a human transcript map *Nat. Genet.* **1995**, 10, 369-371.

6. National Center for Biotechnology Information: http://www.ncbi.nlm.nih.gov/UniGene/, 2002.

7. Thompson, J. D.; Higgins, D. G.; Gibson, T. G. Clustal W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids. Res.* **1994,** 22, 4673-4680.

8. Felsenstein, J. Phylip-phylogeny inference package. *Cladistics* **1989**, 5, 164-166.

Chapter IV. Discussion

I. Discussion

Today's success in bioinformatics and cheminformatics studies depends largely on the integration of hundreds of sources that usually consist of many different data types (numerical, text, structural data). Data reading components of the data pipelining system are a group of components, each of which is capable of reading or importing Oracle database records, delimited flat files, SD molecule records, etc. respectively. Thus, data sources of different formats can be fed into a single data pipeline, and further analyzed by merging, sorting, data modeling and other components. The data pipelining technology allows the most data sets of research interest to be imported into the pipeline and processed according to the user specified computational network protocol, thus offering virtually unlimited flexibility. With this new freedom, users will be able to evaluate or process multiple data sets (or multiple databases) as if they were within a single source.

The extremely high performance of data pipelining makes it ideally suitable in a high-throughput research environment [1]. By creating a computational protocol to capture the human expertise of acquiring, analyzing and managing data, the raw data can be automatically processed with the data pipeline as the data is generated. The results of a data pipeline can be ultimately stored in a database. Served as the complement to the database system, data pipelining can be used to clean, evaluate and compare contents of databases.

Data pipelining can be used to virtually screen a compound database to discover drug candidate leads. Creating a new medicine is a complex business, costing over $300 million and typically taking between 12 and 15 years per marketed drug. Drugs available

in the market have passed rigorous scientific and regulatory hurdles. From the hundreds of thousands of potential drug compounds that are evaluated against disease targets, only a handful meet the requirements necessary to progress through the full drug approval process. It is estimated that at least 10,000 compounds are evaluated for each drug that is approved for use in the United States. Data pipelines in Figure 3.1 were constructed to predict the dehalogenation rates of chemical compounds. If we replace the dehalogenation rate constant (Y2) with the IC_{50} as an activity of a compound, which is the concentration of the compound required to produce a standard response to a particular disease target in a given time, such pipelines can be applied to calculate IC_{50} values for compounds in a chemical library. Thus, library compounds can be prioritized according to their calculated IC_{50} values, and the top ranked compounds can be experimentally tested first to discover drug candidate leads more quickly. By examining only the highest scoring compounds, experimentally testing expenses can be greatly reduced.

The data pipeline in Figure 3.5 allows a user to identify a core sequence to represent a cluster and find the 95% confidence distance interval. What a user needs to do is only to construct the data pipeline once and configure each component appropriately. Without the data pipelining technology, a user may have to accomplish tasks in several steps, including using Clustal W to do multiple sequence alignment, applying Phylip to calculate distance matrix in Unix, using some statistical software package in Windows to identify core sequence and construct the 95% confidence interval, and transferring files between different operating systems. Such work is not efficient and the user is likely to make mistakes. By utilizing the data pipelining technology to perform tasks automatically, these problems can be avoided.

With components for data modeling, molecular modeling, bioinformatics and other functions, a data pipeline is capable of identifying trends, discovering relations and patterns among the data running through it, thus allowing data to be converted into useful knowledge.

II. Conclusion

The data pipelining approach is designed to address the challenges of scalability, data integration and data mining in today's data-rich environment. By guiding the flow of data through a network of modular computational components, data pipelining provides great flexibility and fine control over analysis.

One of the limitations of the data pipelining technology is that the number of computational components that can be provided is limited. However, the scope and topics of research problems in bioinformatics and cheminformatics are changing everyday. In the case where there are no appropriate computational components for a specific problem a user is attempting to solve, the open architecture of a data pipelining system should allow the user to implement code and construct components by himself.

References

1. Tozer, J. R. Ask More of Your Discovery Data. *Genomics & Proteomics* **July/August 2001**, 65-68.