UNIVERSITY OF CALIFORNIA

Los Angeles

Development of Statistical Online Computational Resources and Teaching Tools

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Science

in Statistics

by

Dushyanth Krishnamurthy

2005

The thesis of Dushyanth Krishnamurthy is approved.

Ivo Dinov

Rob Gould

Yingnian Wu

Jan de Leeuw, Committee Chair

University of California, Los Angeles

2005

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Ivo Dinov, for having given me the opportunity to work with him. His advice and guidance has helped during my study here. I am also grateful toward the university and my advisor for having supported me financially throughout my stay at UCLA. I owe my committee a special thanks for their contribution to this thesis. Finally I would like to thank my family for their help and support throughout my stay at UCLA.

ABSTRACT OF THE THESIS

Development Of Statistical Online

Computational Resources And Teaching Tools

By

Dushyanth Krishnamurthy

Master of Science in Statistics
University of California, Los Angeles, 2005
Professor Jan de Leeuw, Chair

The advent of technology has facilitated computer based techniques for statistical analysis of data. In addition, the Graphical User Interface (GUI) has enabled the development of highly interactive programs. Statistical Online Computational Resources (SOCR) is a hierarchy of portable online interactive software that can be accessed through the internet. It combines statistical data analysis tools with interactive demonstrations, simulations, and GUIs designed to supplement methodological training in statistics classes.

In this thesis we present three categories of tools that form part of the SOCR online repository. For fitting data to probability distribution models, the SOCR *Modeler* is employed. Features are provided for loading data, fitting various models, and checking the goodness of fit using visual and quantitative results. An example is

provided to show the application of the Mixed Modeler tool to a medical imaging application.

Next, the SOCR *Analysis* tool provides an interface for performing statistical analysis of data. We present a general framework for the SOCR *Analysis* tool and study the implementation of Aanalysis of Variance (ANOVA) as a specific type of analysis tool.

To improve the understanding of statistical concepts, SOCR *Games* provides a set of interactive software with illustrations. We present a demonstration of Fourier and wavelet transforms as part of the SOCR *Games* package. The effect of data compression can be explored visually for various wavelet transforms.

1. INTRODUCTION

1.1 Statistics Online Computational Resources

The Statistics Online Computational Resource (SOCR) seeks to compile a hierarchy of portable online interactive aids for motivating, modernizing and improving the teaching format in college-level probability and statistics courses. This includes a repository of software tools, instructional materials and online tutorials. The SOCR resources allow instructors to supplement methodological course material with hands-on demonstrations, simulations, interactive graphical displays illustrating in a problem-driven manner the presented theoretical and data-analytic concepts. In this thesis we designed a number of applets, user interfaces and demonstrations, which are fully accessible over the internet as part of the SOCR project.

1.2 Design & Implementation

The primary contribution of this thesis is the design of the software modules that would allow flexibility in addition of statistical experiments and the design of intuitive GUI's to help understand statistical concepts. SOCR is designed as a set of stand alone applets, each demonstrating a particular topic in statistics. Each applet is composed of a top level component that is responsible for the display and manipulation of data. In addition each applet contains a number of sub components that are responsible for a particular experiment or demonstration. This design allowed flexibility in adding new demos and

experiments while saving time by grouping the common functionalities into the higher level component.

1.3 Organization of the thesis

The remainder of the thesis is organized as follows. We examine modeling of probability distribution functions using datasets in section 2. The design for the software package is discussed followed by examples of actual probability distribution modelers and their implementation. In section 3 we examine the implementation of the Analysis package. We present Analysis of Variance (ANOVA) and Regression Analysis at two examples of the Analysis tools package. In section 4 we discuss the implementation of wavelet transforms and Fourier analysis as part of the games package. Finally we discuss some applications of SOCR and suggest some future developments.

2. PROBABILITY DISTRIBUTION MODELING

In statistics, we are often given a sample dataset that is known or assumed to be generated from some probability distribution. We are then required to form statistical inferences regarding the nature of the probability distribution that generated the samples. In some cases the type of distribution will be known and the parameters of the distribution will be unknown. In other cases both the type of distribution and the parameters will be unknown. In both cases there will be parameters associated with the distribution function that need to be estimated.

For many distributions the parameter of interest will belong to some known set Λ. Given a set of observations from the distribution we will be able to estimate the parameter from the observations using statistical techniques that either minimize some criterion or have certain properties such as producing unbiased parameter estimates. Estimation techniques could be closed form solutions or iterative procedures. Details of parameter estimation techniques can be found in [1,2]. If the distribution is known *a priori* the problem is simply one of estimating the parameters associated with the distribution, assuming that the parameters are estimable. In other cases where the data comes from a completely unknown distribution we could try fitting a number of distributions to the data and use statistical tests and visualization to test the goodness of the model fit.

The SOCR Modeler is a useful resource for univariate distribution modeling problems and provides the following features:

- An easily accessible online system for fitting various probability distribution models to data.
- Data in the from of a histogram or as raw data.
- A Graphical User Interface (GUI) for visualization of the model fit.
- Data generation techniques for generating random samples from common distributions.
- Goodness of fit test for checking quality of fit.

In the following sections we discuss the design of the software module followed by some examples of actual probability distribution modelers.

2.1 Design of SOCR Modeler

The SOCR Modeler was designed with a some key features in mind such as interactive GUIs, easy data manipulation and the ability to add new modeling plug-ins as they became available.

The SOCR modeler design is shown in Figure 1. It consists of a core module that is responsible for the common functionality of the modelers, and plug-ins that implement functionality specific to each modeler such as parameter estimation and result generation.

Core Components

The bulk of the code is contained in the core. There are four functional modules within the core including the GUI components, data management module, modeler interface and

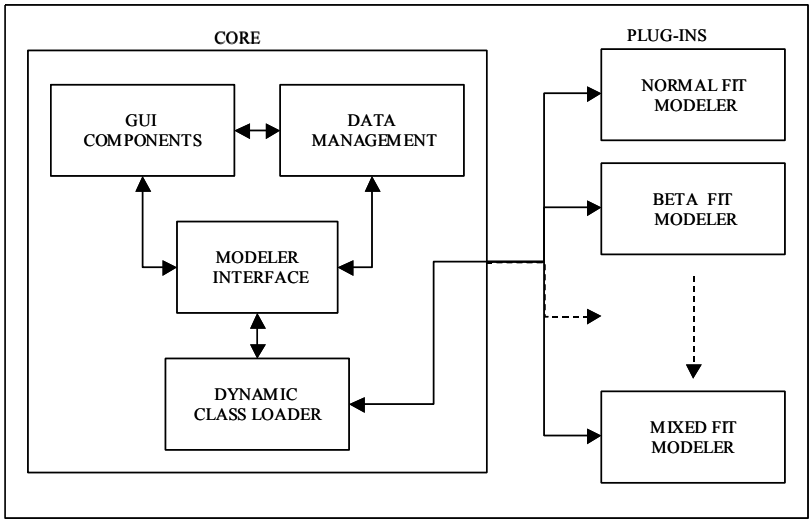

Figure 1. Block diagram of SOCR Modeler showing functional components.

the dynamic class loader. Among these the first two are highly interrelated and work together closely to synchronize the visualization of the data.

Modeler GUI: The GUI consists of three regions as shown in Figure 2. The left pane

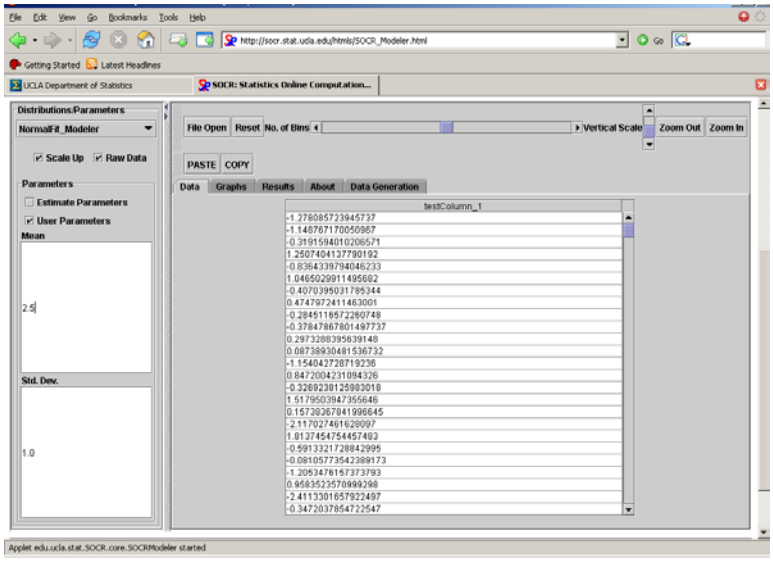

Figure 2. Screen shot of SOCR Modeler GUI.

displays the available modelers and the parameters associated with each modeler. The top portion contains a common toolbar and the remaining portion consists of tabbed panes that display the inputs and outputs. The first tab (Data) displays a table for entering data either as histogram data or as raw data. The second tab (Graphs) is shown in Figure 3 and displays the graph used for inputting and displaying the histogram of the data. The model fit is displayed as a red line superimposed over the histogram. The probability model can be plotted in its original scale or scaled up proportional to the histogram. The third tab (results) is shown in Figure 4. and displays the results of the model fit. The fourth tab

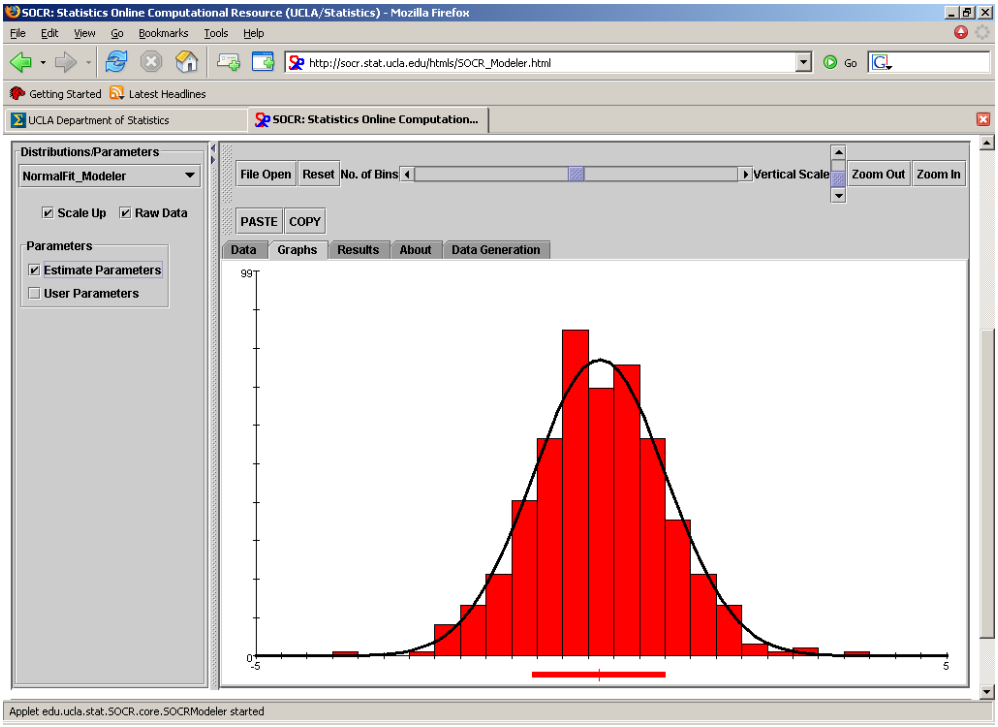

Figure 3. Illustration of SOCR Modeler graph showing normal distribution model fit.

(About) displays help information about the modeler and is currently under development. The final tab (Data Generation) allows data generation and is discussed later in this section.

Data Management: Data management consists of the Input/Output methods for the data and the corresponding data objects. Data input can take place through the keyboard, mouse, file or from the system clipboard. The GUI objects used for representing and manipulating data are the table and the graph tabs (Figure 2 and 3).

Data is entered directly into the keyboard either as raw data or as a histogram. In addition, tab separated text files can be loaded using the FILE OPEN button. Data can be

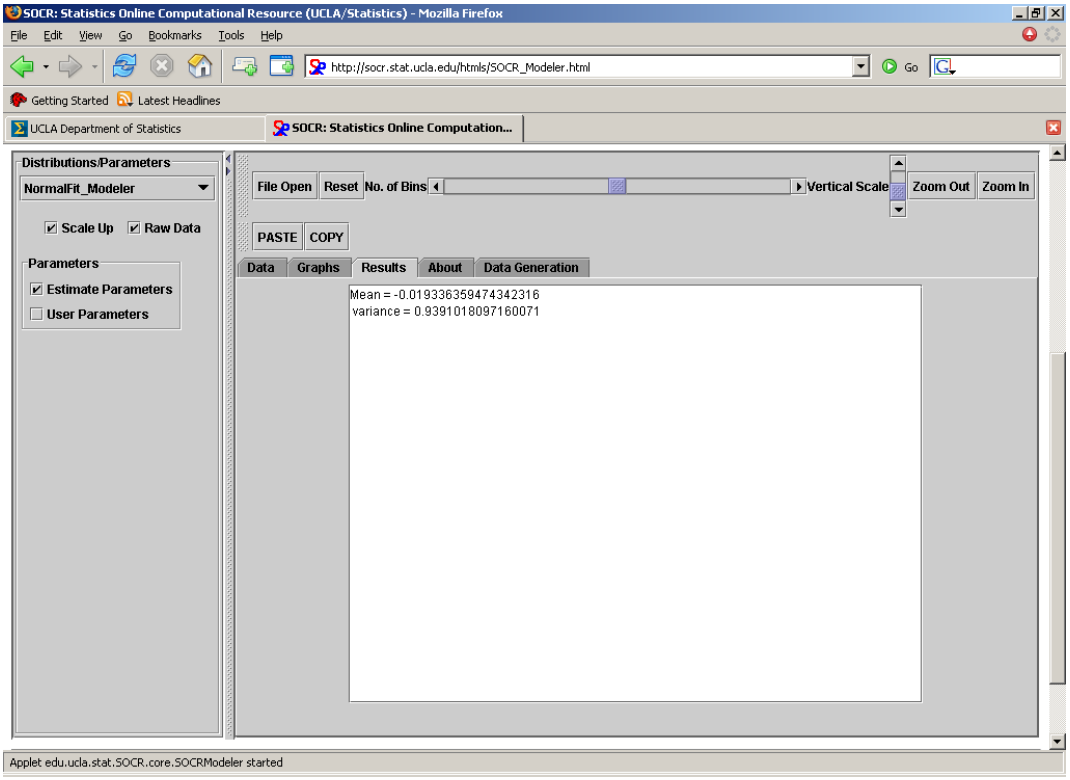

Figure 4. Example showing SOCR Modeler results panel.

copied from applications such as excel and pasted into the table using the PASTE button. Also, clicking the mouse on the graph panel changes the levels of the histogram.

There are three data objects for storing the data internally. One for storing the raw data values, a second for storing the histogram values and a third for storing the values estimated from the model fit. When a change is made to the data using one of the input methods discussed above, a synchronization mechanism is triggered that converts data from the raw format to the histogram format or vice versa depending on the input mechanism. Also, changes to the data recalculates the model fit using the new data.

Modeler Interface: The modeler interface is a java class that behaves as an intermediary between the core and the implementations of the modeler plug-ins. The core components send their request for data or calculations to this interface. The interface then passes on this request to one of the modelers based on the current selection. For example, if the core component requires the fitted model values, it requests this from the interface. If the currently selected modeler is the normal fit modeler, the interface will send this request to the Normal Fit Modeler which then returns the parameter estimates to the core via the interface. In addition, the interface specifies certain standards that must be followed by all the modeler plug-ins.

Dynamic Class Loader: When a user selects a specific modeler through the GUI, it must be loaded at run time and an association formed between the interface and the selected modeler plug-in. This functionality is taken care of by the dynamic class loader.

The class loader along with the interface serve to separate the specifications from the functionality of each plug-in. This allows easy addition of new probability modeler plug-ins as long as they conform to certain standards specified by the Modeler Interface. This also allows the addition of new modelers without recompilation of core components.

SOCR Plug-ins: The SOCR plug-ins perform parameter estimation and provide model values. The SOCR plug-ins need to implement the specifications set by the Modeler Interface for data input and output. They are also given access to the left pane of the GUI (Figure 2 - left) for getting user input. All probability modelers must have functions that takes raw data as input, estimates its parameters and returns the fitted model values.

2.2 Implemented Modelers

Each model is provided with univariate input data and returns a probability distribution function (pdf) model after performing an estimate of the parameters associated with the model. In this section we discuss some examples of modelers that have been implemented as part of the SOCR Modeler package.

Normal Modeler

The normal distribution is one of the most commonly used distributions that arises in a number of applications. Details can be found in [3,4]. The distribution is completely specified by its mean μ and standard deviation σ. The pdf of a normal random variable is given by

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \tag{1}$$

Given a data set $Y_1,\ldots,Y_N$ we obtain point estimates of the parameters from the data using the sample mean and standard deviation using the formula

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} Y_i \tag{2}$$

The standard deviation can be estimated as

$$\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{N}(Y_i - \hat{\mu})^2}{N-1}} \tag{3}$$

Equation (3) gives an unbiased estimate of the true standard deviation.

We provide the user with the option of setting the mean and variance of the fit manually or allowing the system to estimate the parameters for the models using equations (2) and (3). The resulting fit is super imposed over the histogram of the data. Figure 3 shows an example of a normal fit where the parameters have been estimated. Figure 5 shows the same data with a user defined mean and variance.

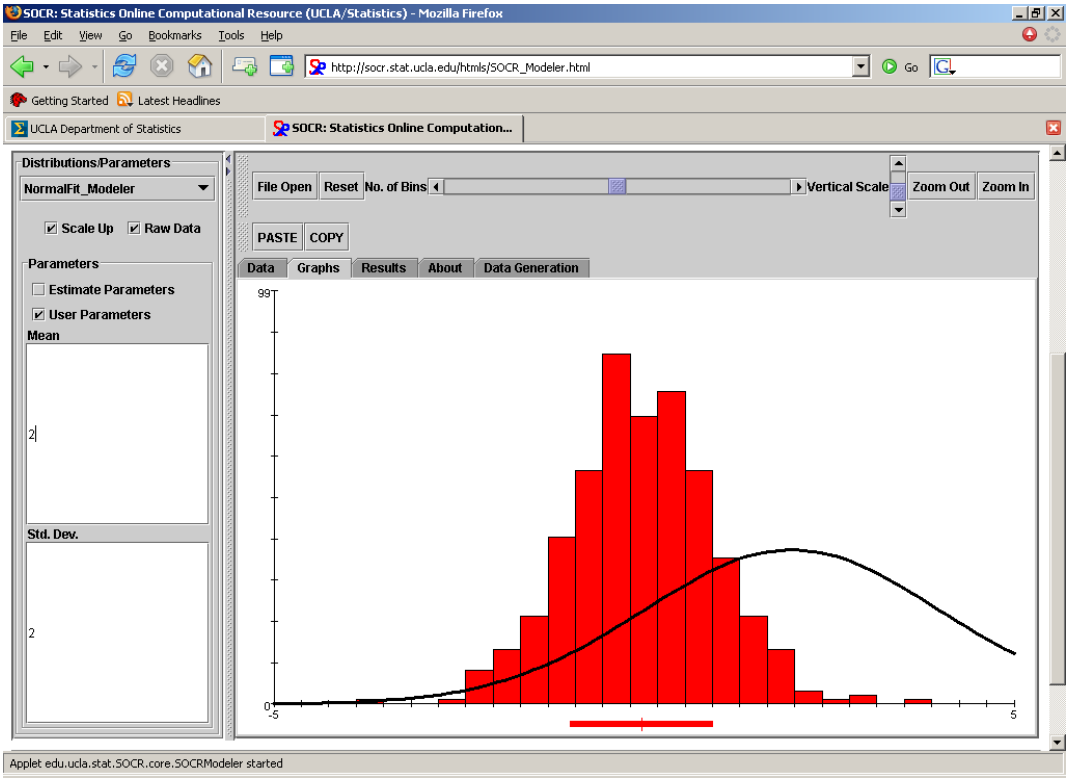

Figure 5. Example of normal distribution fit with user specified input parameters.

Exponential Modeler

The exponential distribution has only one parameter called the rate parameter λ. The pdf is given by

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & otherwise \end{cases} \tag{4}$$

where λ is greater than zero. It is often used in applications to model the successive rate of occurrences of events such as arrivals of customers at a facility.

Given a data set $Y_1,\ldots,Y_N$ we can obtain point estimates of the parameter from the data using the closed form equation

$$\hat{\lambda} = \frac{1}{\hat{\mu}} \tag{5}$$

where $\hat{\mu}$ is the sample mean. Figure 6 shows an example of an exponential model fit.

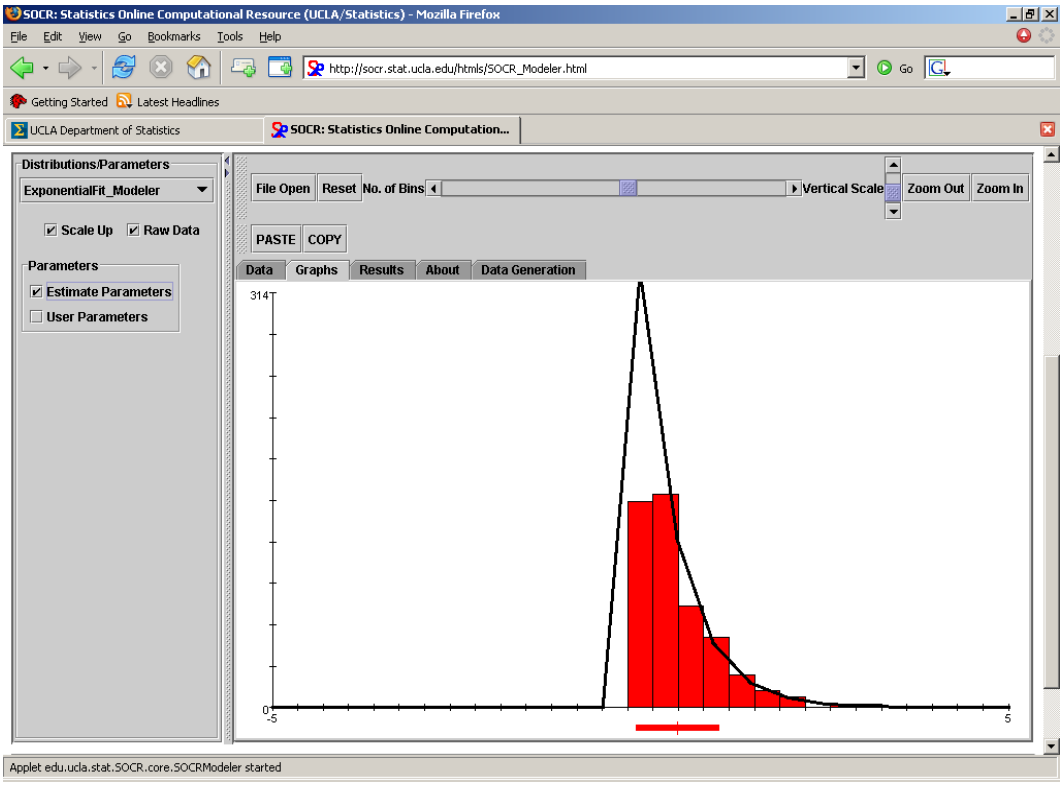

Figure 6. Example of exponential distribution model fitted using estimated parameters.

Mixed Modeler

In certain applications there are measurements consisting of data from multiple sources, each source having a different distribution, different parameters or both. In these applications there is an additional latent variable associated with each data point

indicating the source it belongs to. A common example is measurements of heights from a population of males and females. This can be modeled as a mixture of two normal variables. In this case the latent variable associated with each measurement is the sex of the species. In addition, there is also a variable that measures the proportion of males and females in the population. The latent variable is not always measurable and in these cases they need to be estimated along with the other parameters.

In this section we describe the implementation of a Mixed Modeler that fits a n normal mixture model to a given data set. The algorithm is implemented using the expectation maximization (EM) algorithm that performs an iterative search to find an optimal solution for the unknown parameters. We start by describing the normal mixture model, derive expressions for the Maximum Likelihood estimate of the unknown parameters and finally discuss the implementation of the EM algorithm for estimating these parameters.

Normal Mixture Model: A normal mixture model X is a continuous random variable that consists of n normal random variables. The pdf of the normal mixture model is given by

$$P_X(x) = \sum_{j=0}^{n-1} \lambda_j X_j \tag{6}$$

where

$$X_j = \frac{1}{\sqrt{2\pi\sigma_j^{\,2}}} e^{-(x-\mu_j)^2/2\sigma_j^{\,2}} \tag{7}$$

and $\sum \lambda_j = 1$. Hence the normal mixture can be described as a sum of weighted normal random variables where the λ_j's denote the weight of each normal component. We can denote the set of parameters associated with the normal mixture by

$$\Theta = \begin{pmatrix} \lambda_0 \\ \mu_0 \\ \sigma_0 \\ \vdots \\ \lambda_{n-1} \\ \mu_{n-1} \\ \sigma_{n-1} \end{pmatrix} \tag{8}$$

Data Set: Given a set of N measurements Y_i from a normal mixture distribution, we can define a set of N latent variables $Z_i, i = 1,\ldots,N$ associated with each Y_i taking values $0,\cdots,n-1$ representing the normal component of the mixture that data Y_i belongs to. Given latent variables Z_i, we can define the pdf of Y_i by

$$P_{Y_i}\left(y_i; Z_i = j, \mu_j, \sigma_j\right) = \frac{1}{\sqrt{2\pi\sigma_j^{\,2}}} e^{-(x-\mu_j)^2/2\sigma_j^{\,2}} \tag{9}$$

ML Estimates: In this section we derive ML estimates for the parameters in the vector Θ. Let us define a set of variables $l_0,\ldots,l_{n-1}$ where l_i is the number of latent variables equal to i. We can form a vector from the observed values denoted as Y form a vector of latent variables denoted by Z. The ML estimate of the parameter vector Θ is then given by

$$\hat{\Theta}_{ml} = \arg\max_{\Theta} \ln P(Y;\Theta) \tag{10}$$

The values of Θ that maximizes the above expression can be calculated by differentiating the function with respect to Θ and equating the result to zero as shown in equation ().

$$\frac{\partial \ln P(Y;\Theta)}{\partial \Theta} = 0 \tag{11}$$

The likelihood function in this case is given by

$$L(\theta) = \lambda_0^{l_0} \cdots \lambda_{n-1}^{l_{n-1}} \prod_{\{Y_i:Z_i=0\}} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-(Y_i-\mu_0)^2/2\sigma_0^2} \quad \cdots \quad \prod_{\{Y_i:Z_i=n-1\}} \frac{1}{\sqrt{2\pi\sigma_{n-1}^2}} e^{-(Y_i-\mu_{n-1})^2/2\sigma_{n-1}^2} \tag{12}$$

Taking log of the likelihood function, differentiating the result with respect to the parameters in Θ, and equating to zero yields the following ML estimates for the weights of the normal components as

$$\lambda_i = \frac{l_i}{N} \tag{13}$$

Similarly we can also derive expressions for the estimates of the mean as

$$\hat{\mu}_k = \frac{\sum_{\{i:Z_i=k\}} Y_i}{l_k} \tag{14}$$

and for the variance as

$$\hat{\sigma}_k^{\,2} = \frac{\sum_{\{i:Z_i=k\}} (Y_i - \hat{\mu}_k)^2}{l_k} \tag{15}$$

E-M Algorithm: When the latent variables Z_i is known, the estimation of parameters associated with each normal component is straightforward and can be estimated using equations (13) to (15) above. In other cases when Z_i is unknown or missing, equations (the ones just above) cannot be directly used to obtain estimates of the parameters. In these cases the $Z_i's$ need to be estimated along with the parameter vector Θ. We can rewrite equation (11) as

$$\frac{\partial \ln P(Y/\Theta)}{\partial \Theta} = E_{Z\sim P(Z/Y,\Theta)}\left\{\frac{\partial}{\partial \Theta}\ln P(Y,Z/\Theta)\right\} \tag{16}$$

and equate to zero. The estimation of Z in the above equation is called the E-step of the algorithm and is an expectation operation. Finding parameter estimates that maximize $P(Y,Z;\Theta)$ is called the M-step. In the M-step of the algorithm we differentiate the term inside the expectation by equate it to zero. The EM algorithm works in following way.

1. An initial estimate of the parameter vector called Θ_0 is provided to the algorithm.
2. The estimate of the parameter Θ_t at the t th iteration is used in calculating the expected value of each Z_i (E-step).
3. The estimates of $Z_i^{(t)}$ and Θ_t are used to obtain the values of Θ_{t+1} using equations (13) to (15) (M-step).

The EM algorithm converges to a solution by alternating between the E-step and the M-step. The initial estimate Θ_0 needs to be close to the true parameter values to ensure the algorithm converges to the global maximum. Figure 7 shows the result after forty iterations when the algorithm has converged to a local maximum. Figure 8 shows convergence of the same data set with different initial values. We see that the solution has converged to the global maximum. Details of EM algorithm can be found in [5] and [6].

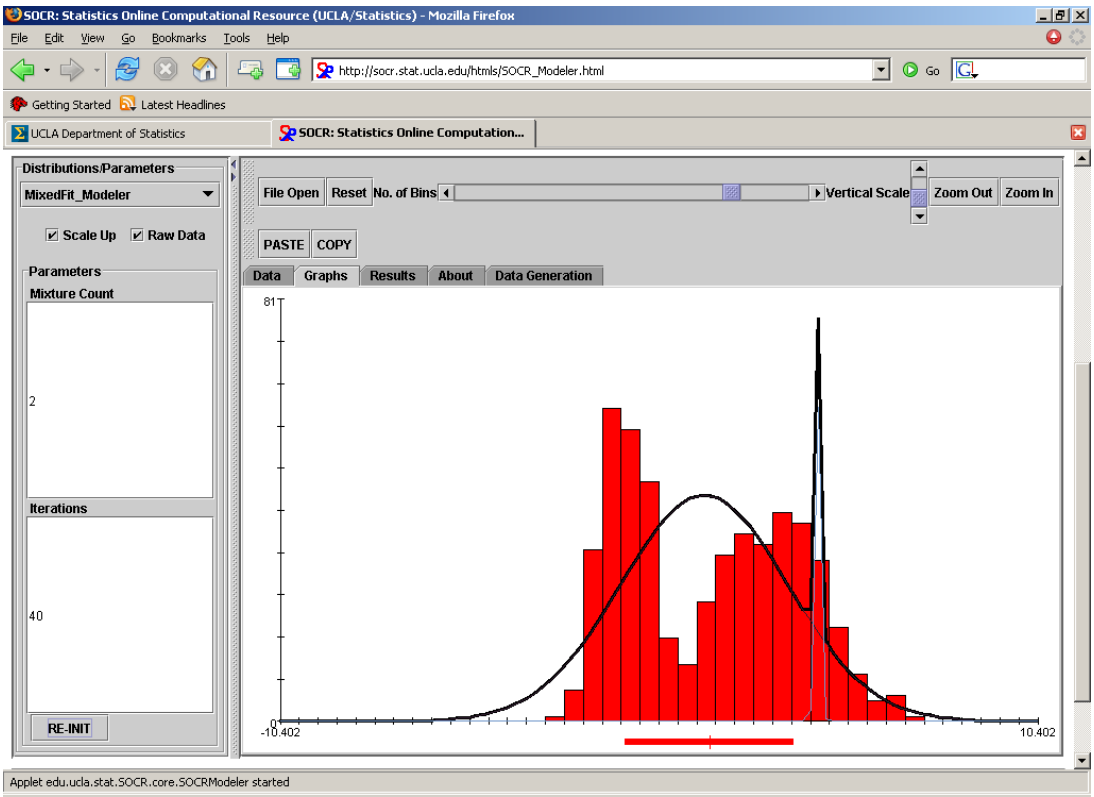

Figure 7. Example of mixed model fit with two components showing convergence to local maximum.

2.3 Goodness of Fit

Once the model parameters have been estimated we need a method for testing how well the model fits the data. Such a measure can be obtained by using the chi square Goodness-of-Fit test. An example of the chi-square distribution can be found in [7] and details of the test can be found in [3,8]. This testing method is suitable only for categorical data. Hence we split the data into categories based on the histogram of the data. Given n observations, and k categories, we can denote the number of observations in each bin by n_i where $i = 1,\ldots,k.$ We denote the probability of a data point occurring in the ith bin by p_i. This can be calculated from the pdf of the distribution. The expected number of data points in the i^{th} bin is then given by np_i. The chi-square statistic measures the deviation of the expected number of data points from the observed number of data points. The chi-square test statistic is then given by

$$\chi^2 = \sum_{i=1}^{k} \frac{(n - np_i)^2}{np_i} \tag{17}$$

The chi-square statistic is compared with a chi-square distribution with $(k-1)$ degrees of

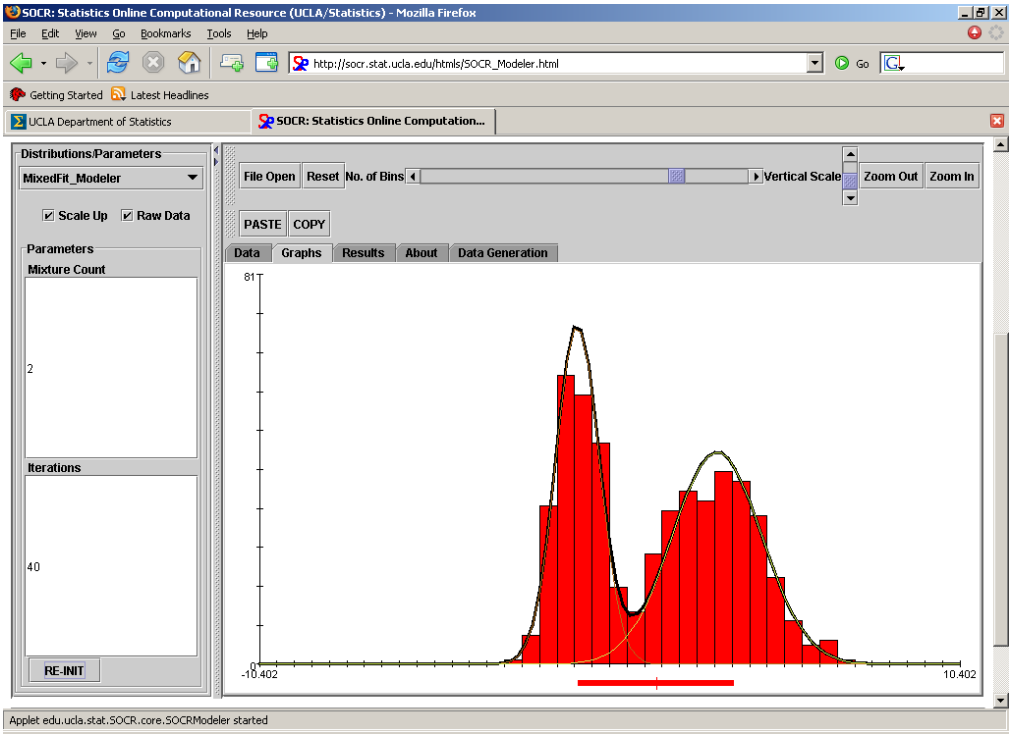

Figure 8. Example of mixed model fit with two components showing convergence to global maximum.

freedom and a p-value is calculated based on the area to the right of χ^2. This test does not depend on the distribution and can be implemented easily for various distribution modelers but the test results depends on the number of bins in the histogram and may vary accordingly.

2.4 Data Generation

The SOCR Modeler has the provision to generate sample data from various distributions. This is helpful as a teaching aid and can be used to demonstrate the fitting of the models using generated samples. Figure 9 shows the data generation tab. The panel allows one to

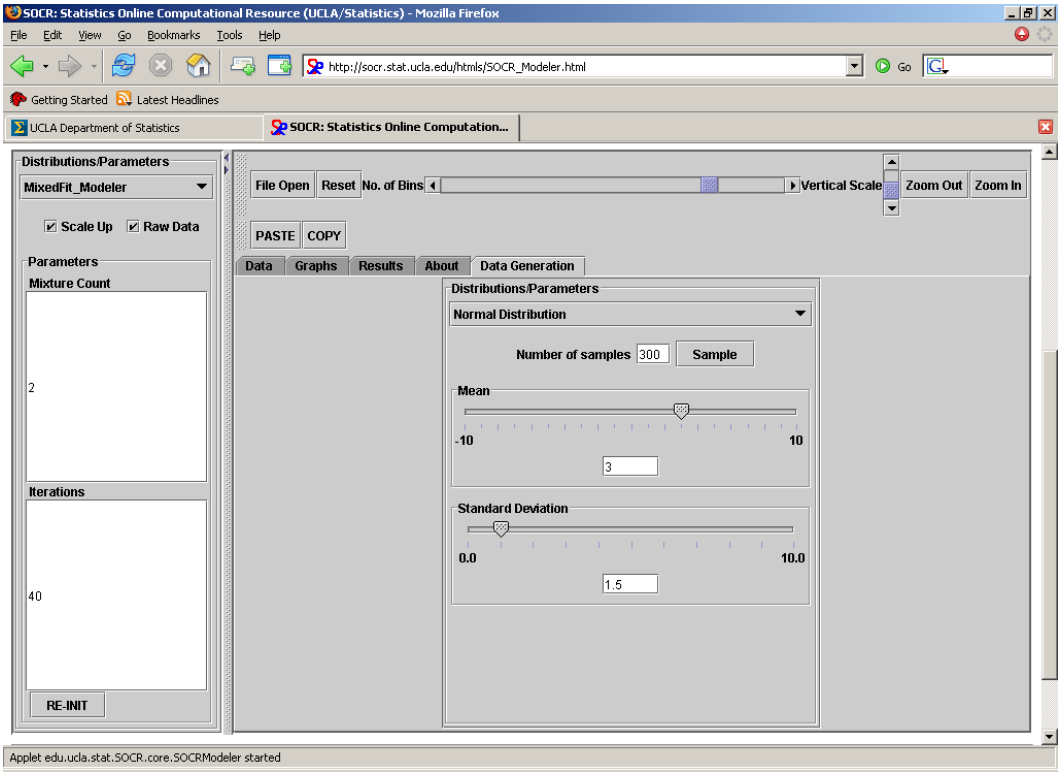

Figure 9. Screen shot of data generation panel of SOCR Modeler.

select a distribution, set the parameters and select the number of samples required. The data is then generated as a pseudo random sequence from the distribution and placed in the table.

3. STATISTICAL ANALYSIS

Statistical analysis deals with the analysis of sample datasets from a population to arrive at conclusions or inferences about the underlying population. There are a number of statistical analyses methods such as hypothesis testing, regression analysis, analysis of variance (ANOVA), analysis of categorical data and so on. These tests are quite often performed on data measurements obtained from experimental setups. There are a number of factors in deciding on an appropriate method of analysis depending on factors such as number of populations being analyzed, whether the datasets are numerical, categorical or a mix of both and more importantly on the experimental setup.

The SOCR Statistical Analysis package consists of applets for analyzing quantitative datasets, performing tests on the data and deriving statistical inferences about the data. The SOCR Analysis tool provides a graphical user interface for analyzing datasets as well as examples for students to gain a better understanding of the application of statistical analysis.

3.1 Software Design of SOCR Analysis

The design of the SOCR Analysis module is similar to the SOCR Modeler. Figure 10 shows a block diagram of the SOCR Analysis module. There is a core component that handles the data and GUI functionality but unlike the SOCR Modeler the SOCR Analysis

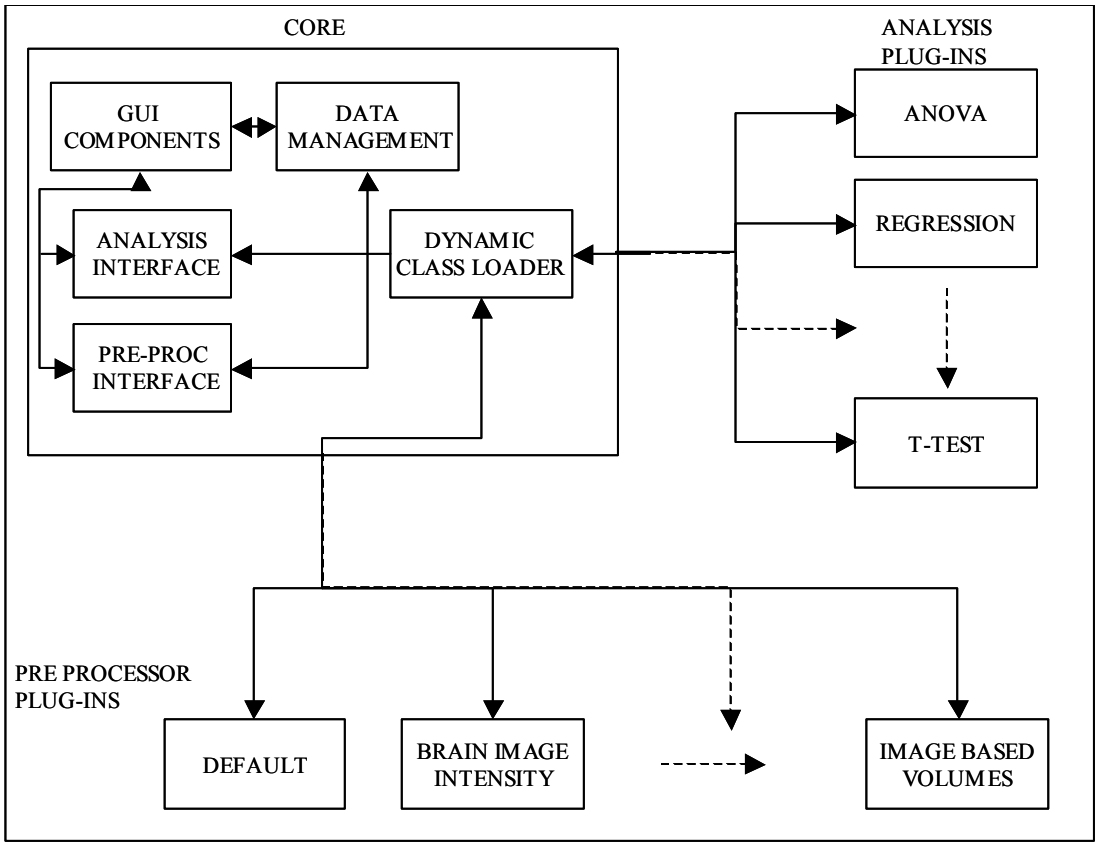

Figure 10. Block diagram of SOCR Analysis showing functional components.

has two sets of plug-ins and interfaces. One set of plug-ins called the Preprocessor plug-ins, handles the pre processing of the data while the other set of plug-ins called Analysis plug-ins, handles the statistical analysis of the data. While the focus of the design in the SOCR Modeler is on the data handling by the GUI components, the focus of the design in the SOCR Analysis is on data formatting and processing.

Analysis GUI: The Analysis GUI is shown in Figure 11. The GUI design is relatively

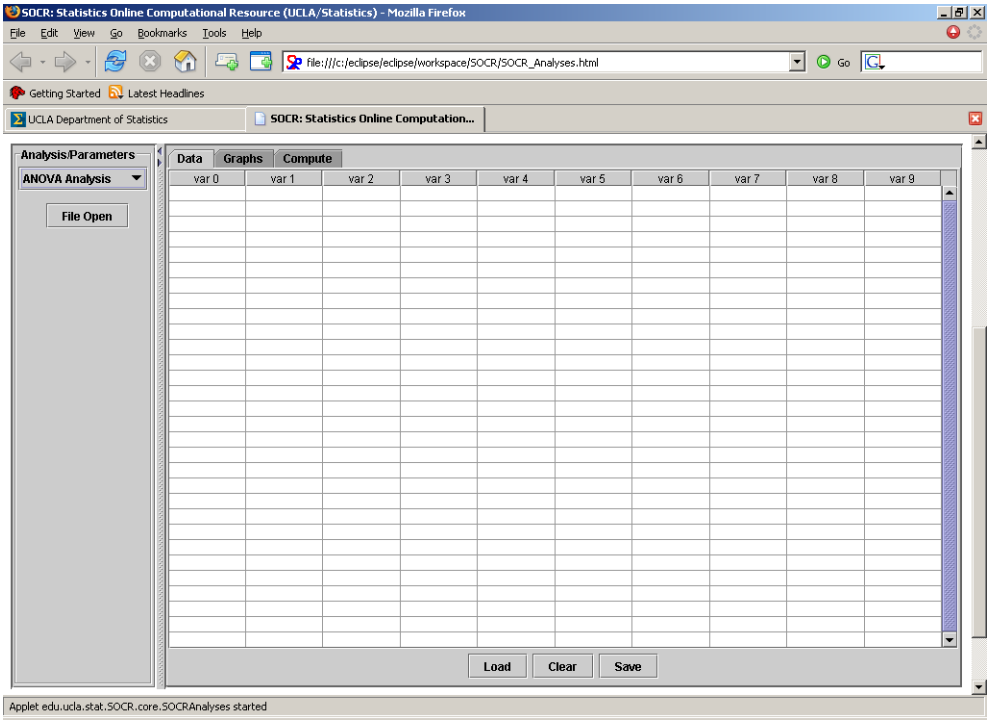

Figure 11. Screen shot of SOCR Analysis graphical user interface.

simple when compared with the SOCR modeler GUI. The toolbar at the bottom contains most of the common functionality. The tabbed panels consists of a table, graph panel, results and mapping tabs. The table is the only input mechanism for the module and can be done either through the keyboard or from a text file. Results are displayed as text in the results panel. The graph panel is available for the analysis plug-ins to display results when available. The mapping panel is shown in Figure 12. The mapping panel can be split into three regions. The top portion is the preprocessing panel. It is responsible for displaying the data formatting and pre-processing options for the data. The center portion

called the analysis panel displays the options for the Analysis plug-ins. The bottom portion displays the filters that can be applied to the data.

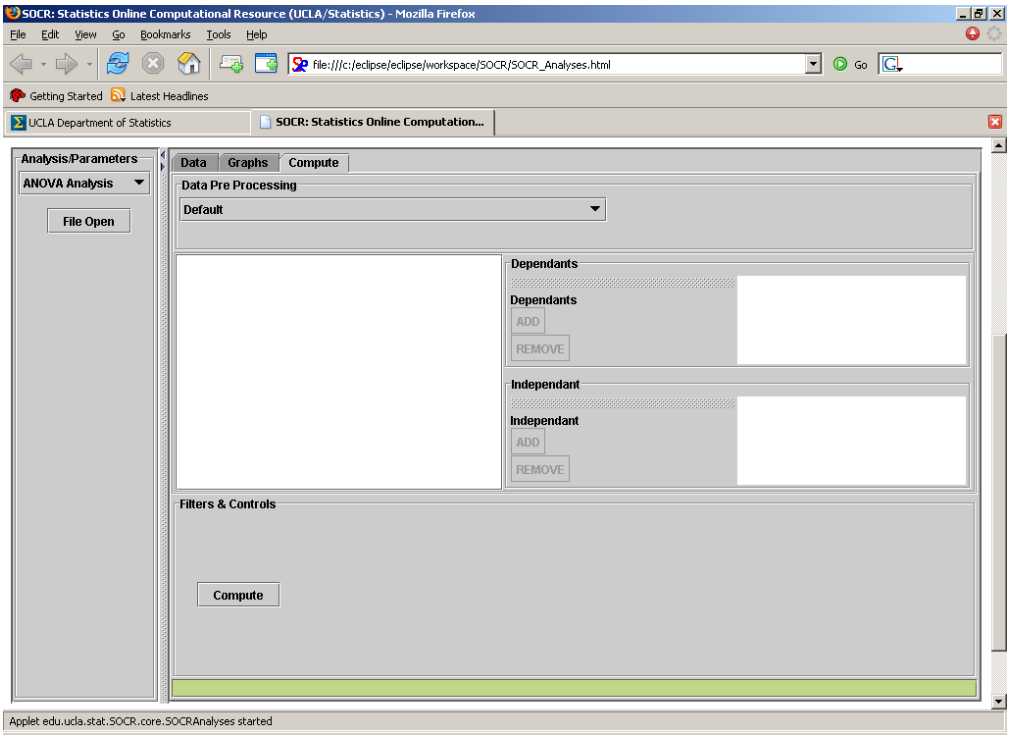

Figure 12. Screen shot showing compute panel of SOCR Analysis.

Data Processing Overview: Data processing and handling takes place in multiple stages both in the core as well as the plug-ins. An overview of the data processing is shown in Figure 13.

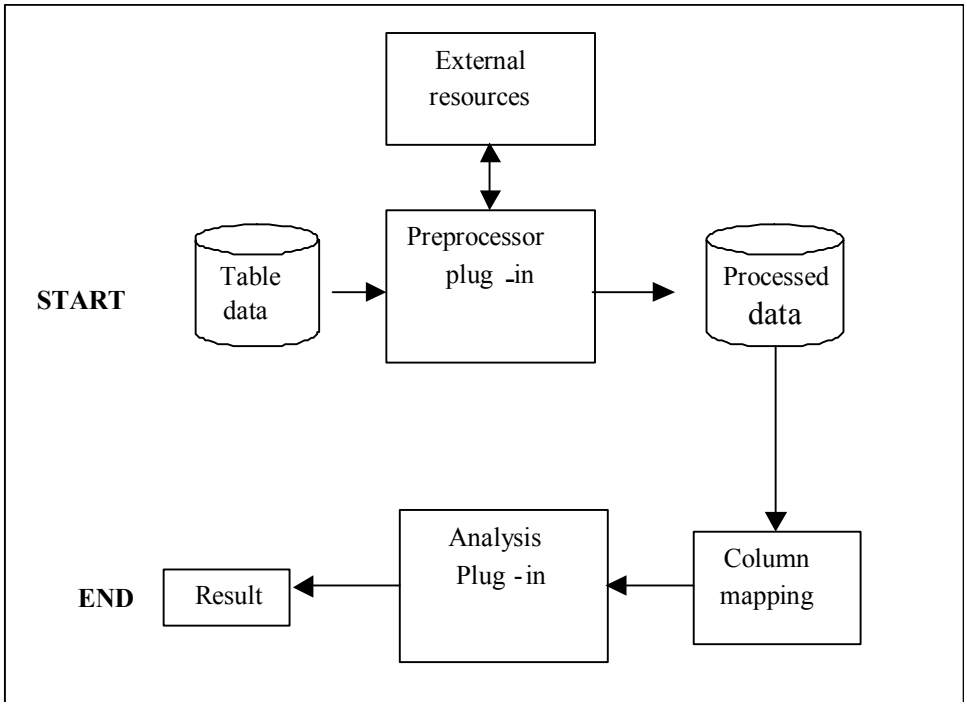

Figure 13.Block diagram of SOCR Analysis data flow process.

The processing of data can be summarized in the following steps.

1. Data is either entered into the table or loaded directly from a file.
2. Once a pre-processor has been selected from the pre-processor panel, the data is passed to the pre-processor plug-in.
3. Depending on the type of preprocessor plug-in selected, the data is processed to yield a new set of columnar data. The pre-processor output is in the form of columnar data. The column headings are displayed in the selection box as available inputs.
4. The user selects some or all the columns for processing by the analysis plug-ins. The number of columns that can be selected depends on the type of analysis plug-

in selected. For example, if ANOVA is selected, the current ANOVA plug-in allows one dependant variable and up to 3 independent variables to be selected.

5. Once the Compute button is clicked the selected columns are passed into the selected analysis plug-in where the statistical analysis is performed and the results sent back to the Core for display.

Pre-processor Plug-ins: The data entered into the GUI table need not be data values. They could be any type of alpha-numeric information that can be understood by the pre-processor plug-in. In addition, the preprocessor plug-in can request user input through the pre-processor panel, and has access to the file system. This gives great flexibility in the design of a pre-processor. For example, one of the table columns could hold image locations and a preprocessor could be designed that reads these image files and extracts information from the images for processing by the analysis plug-ins. The only requirement of a preprocessor is that they have a standard input/output data format.

Analysis Plug-ins: Once the data is converted to the standard form of a number of columns, these columns are displayed as a selection option as shown in Figure 12. The GUI then allows the user to select the appropriate processed data columns depending on the Analysis plug-in settings. Each analysis plug-in performs a standard task such as ANOVA, hypothesis testing etc. The analysis plug-in has access to the results panel and the graph panel of the GUI for displaying the results.

The separation of the statistical analysis from the pre-processing steps is very useful for a number of applications and a good example of software reuse. This allows us to write statistical analysis routines such as ANOVA once and use it many times for different applications by changing the pre-processing steps. Another advantage is the specification of a standard data input/output format that allows seamless integration of the preprocessing and statistical analysis steps. Currently, only a default pre-processor exists that passes the data to the analysis plug-in without any changes. An imaging pre-processor is currently under development.

3.2 Implemented Analysis Tools

In this sub-section we discuss an analysis plug-ins that was implemented as part of the analysis package.

Analysis of Variance

It is a commonly used tool for analyzing categorical experimental data. Details can be found in statistics text books and research papers [3,9]. To perform an ANOVA data consisting of a dependant variable which is numeric and one or more independent variables. For example the dependant variable could be a response or output of a system while the independent variables could be the settings of the system. The results of the ANOVA calculation is displayed in a tabular format (Figure 14).

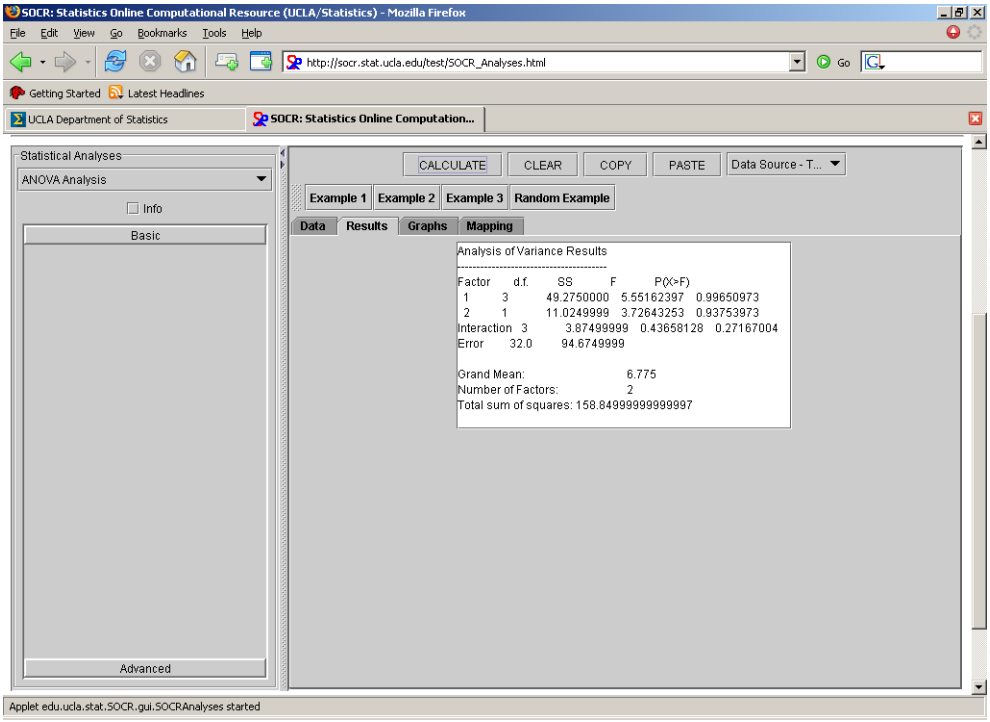

Figure 14. Example of analysis of variance results.

4. GAMES

4.1 Introduction

The SOCR Games package has a number of interactive demos and games for explaining statistical concepts. In this section we discuss the implementation of a Fourier game and a Wavelet game as part of the SOCR Games package. The Fourier Game is a modified version of open source software that has been modified to fit the SOCR Games framework. The wavelet Game builds on the Fourier framework and is capable of demonstrating wavelet transforms. We start this section with an introduction to basis functions, followed by a discussion of Fourier series and a discussion of the implementation and functionality of the Wavelet Game.

Basis Functions

The uses of basis functions arises from the need to represent signals (either real valued or complex valued) as a linear combination of some known signals with certain properties. From a linear algebra perspective this might be accomplished by considering the signal as belonging to a normed linear vector space. If such a space can be found, we might be able to represent the signal as a weighted combination of some basis functions of the vector space. An introduction to vector spaces and basis functions can be found in [10].

Fourier Series

John Baptist Fourier developed the theory of Fourier series as a way of representing periodic signals as a sum of trigonometric series. A signal $x(t)$ is said to be periodic, if for some positive value of T,

$$x(t) = x(t+T) \text{ for all } t. \tag{18}$$

The fundamental frequency for such a signal is given by $\omega_0 = 2\pi/T$. Under certain conditions a periodic signal can be represented by a sum of exponential basis functions that have been shifted and scaled. The exponential basis functions are of the form

$$\phi_k(t) = e^{jk\omega_o t}, k = 0, \pm 1, \pm 2, \ldots \tag{19}$$

The signal $x(t)$ can then be represented by

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_o t} \tag{20}$$

where a_k are the coefficients or the weights of the basis functions. This type of representation is called a Fourier series expansion and is useful in analyzing the spectral content of signals. In addition to being periodic, the function $x(t)$ must satisfy a number of conditions for the Fourier series representation to exist. These are called the Dirichlet conditions and can be found for example in [11]. Assuming that a the representation in equation (20) exists, the coefficients are given by

$$a_k = \frac{1}{T}\int x(t) e^{-jn\omega_o t} dt \tag{21}$$

The Fourier coefficients are complex valued scalars and are given by the inner product of the signal with the basis functions. Since the coefficients are complex we can split them into a phase part and magnitude part. Figure 15 shows a screen shot of the Fourier Game.

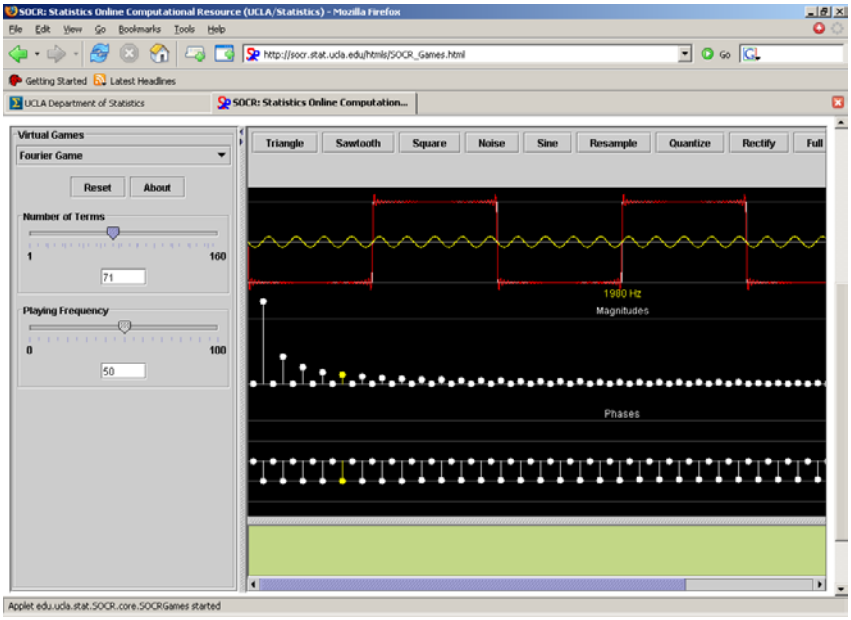

Figure 15. Example of Fourier transform of square wave.

The top portion of the GUI displays the periodic signal. In this case it is a square wave. The middle portion shows the magnitude of the Fourier coefficients and the bottom shows the corresponding phases. The user can adjust the magnitude and phases to see the changes in the signal or redraw the signal using the mouse to see how it affects the Fourier coefficients.

4.2 Wavelet Transforms

Wavelets are scaled and shifted versions of a *mother wavelet* function.. Wavelet transforms decompose a signal down into its basic constituent components but instead of

representing the signal as a series of sine waves of different frequencies the wavelet transform represents signals as linear combinations of wavelet bases. The wavelet basis representation is irregular in shape and uses compactly supported functions. Details of wavelet transforms and their properties can be found in [12].

Implementation

The Wavelet Game applet contains all the code for the GUI components and their functionality (Figure 16). Whenever the game needs to perform Wavelet transforms or inverse wavelet transforms, it passes the data to the wavelet transform interface. Depending on the plug-in that is currently loaded the appropriate transform or inverse wavelet transform is performed. The plug-ins require only two functions.

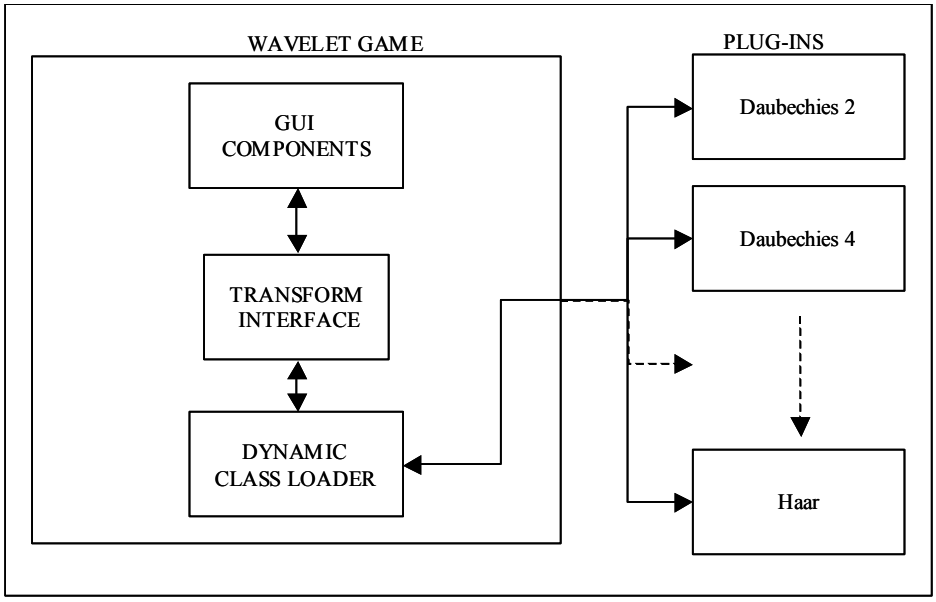

Figure 16. Block diagram of wavelet transform showing functional components.

A transform function that accepts a vector representing a signal and would return a transformed set of coefficients. An inverse transform function that takes a set of transformed coefficients and returns the original signal. This makes addition of wavelet transforms easy.

4.3 Compression Using Wavelets

An application of wavelet transform is data compression. The applet demonstrates the way in which wavelet transform compress data and how certain wavelet transforms work better on certain signals. Compression of a signal can be achieved by taking the wavelet transform of a signal, and discarding a certain percentage of the coefficients by setting their value to zero. Usually the smallest coefficients are discarded. By taking the inverse transform of the coefficients we can study to what degree there is signal distortion. Figure 17 and 18 show wavelet transforms of a sinusoidal wave using Symlett-8 wavelets and Haar wavelets respectively. We see that when 85% of the smallest coefficients have been removed, the inverse transform shows very little distortion in the Symmlet-8 transform whereas visible degeneration in the Haar transform. In contrast when we use a square wave as our signal, we can see that the effects are reversed and the Haar wavelet gives less signal degradation for the same percentage of compression (Figure 19 and 20).

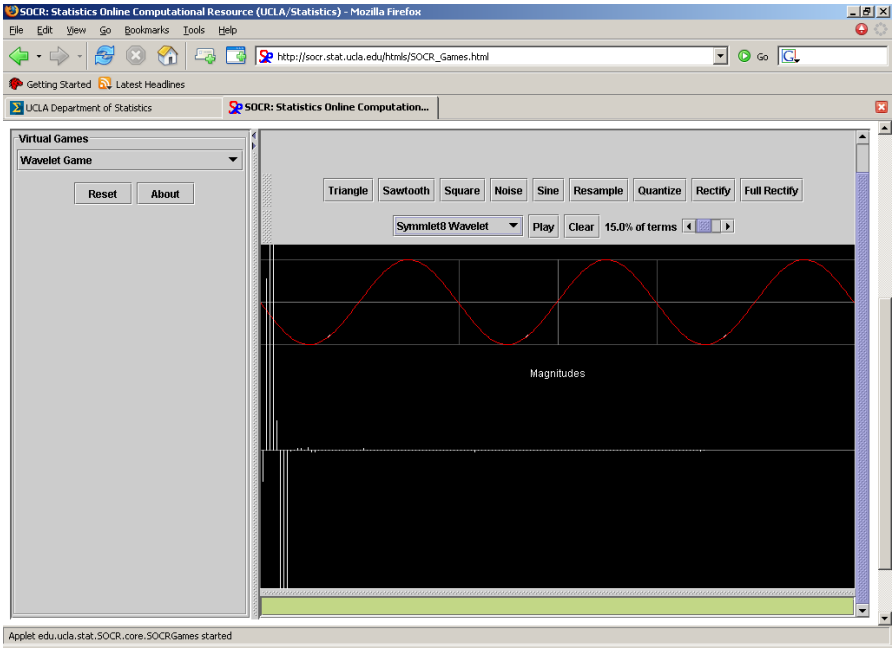

Figure 17. Reconstructed sine wave using Symlet-8 wavelets with 85% compression.

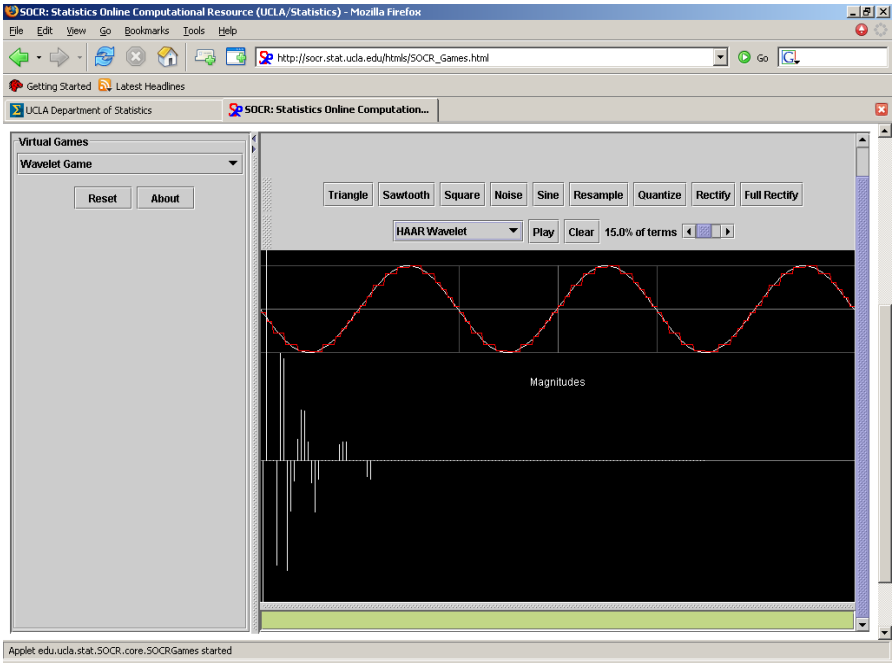

Figure 18. Reconstructed sine wave using Haar wavelets with 85% compression.

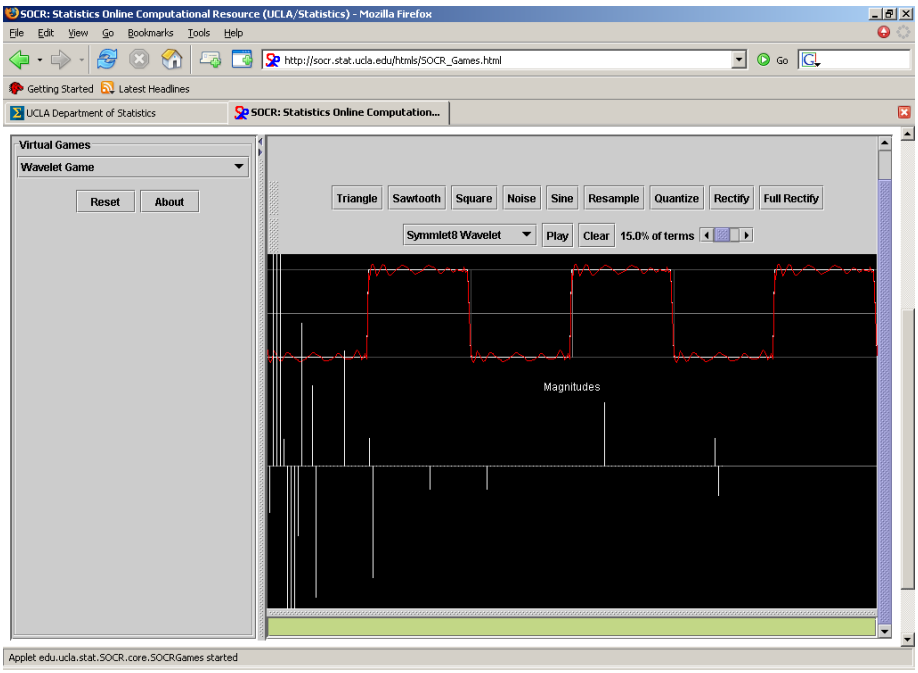

Figure 19. Reconstructed square wave using Symlet-8 wavelets with 85% compression.

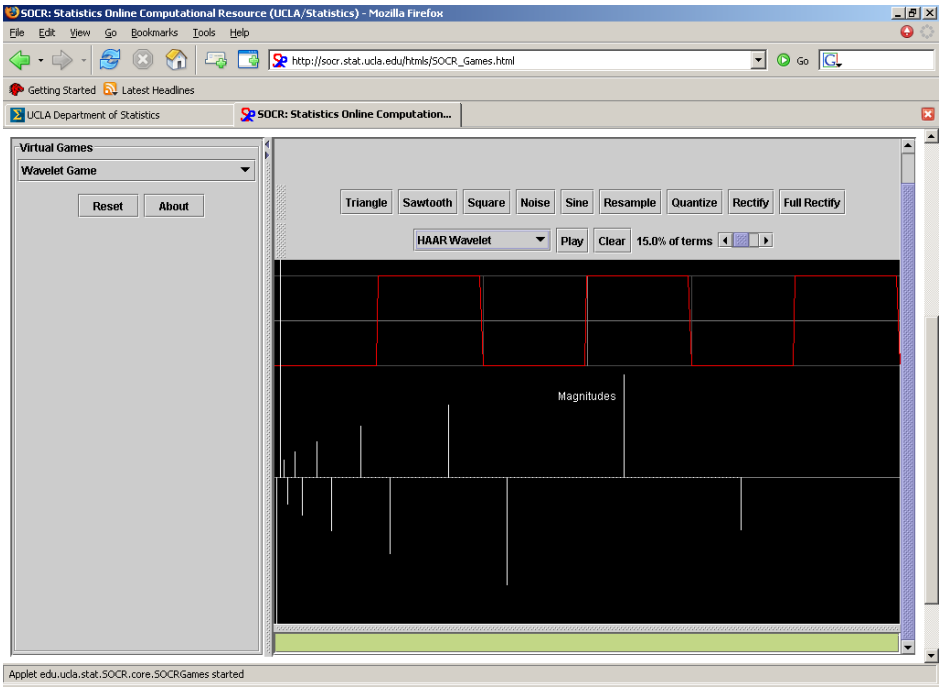

Figure 20. Reconstructed square wave using Haar wavelets with 85% compression.

5. APPLICATIONS

The use of an object oriented design for the packages enhances the applicability of the SOCR modules by allowing easy integration of these tools with other java packages. We discuss an application of the Mixed Modeler fit to an Image analysis application called LONI Visualization Environment (LOVE) that is currently being developed in the Laboratory of Neuro Imaging (LONI).

5.1 Overview of LONI visualization environment

The Laboratory of Neuro Imaging's 3-D viewer, **LOVE**, is a versatile tool for volumetric data display and manipulation of brain images. **LOVE** is a completely portable Java-based software, which only requires a Java interpreter and 64 MB of RAM memory to run on any computer architecture. It allows the user to interactively overlay and browse through several data volumes, zoom in and out in the axial, sagittal and coronal views, and reports the intensities and the stereo-tactic voxel and world coordinates of the data. It also allows the expert users to delineate structures of interest, e.g., sulcal curves, on the 3 cardinal projections of the data. These curves then may be used to reconstruct surfaces representing the topological boundaries of cortical and sub-cortical regions of interest.

5.2 Image Delineation

The brain tissue is composed of white matter, gray matter and Cerebral Spinal Fluid (CSF). It is sometimes of interest to delineate the brain image based on the tissue type. LOVE integrates this functionality into it’s GUI (Figure 21). The user can select circular

regions from the brain for which the algorithm delineates the image based on tissue type. The pixel intensities for a selected region can be modeled as samples from a weighted mixture of three normal density functions where each pixels from each tissue type belong to one of the three normal densities. Since we do not know the parameters of the normal model or the tissue type each pixel represents, we can use the EM algorithm to iteratively estimate the mixed modeler parameters. Once the parameters have been estimated, we can calculate the intersection points of the normal components which serves as hard threshold points to delineate the tissue types based on pixel intensities.

5.3 Algorithm Implementation

The left portion of the LOVE GUI (Figure 21.) shows the selected region of the brain in

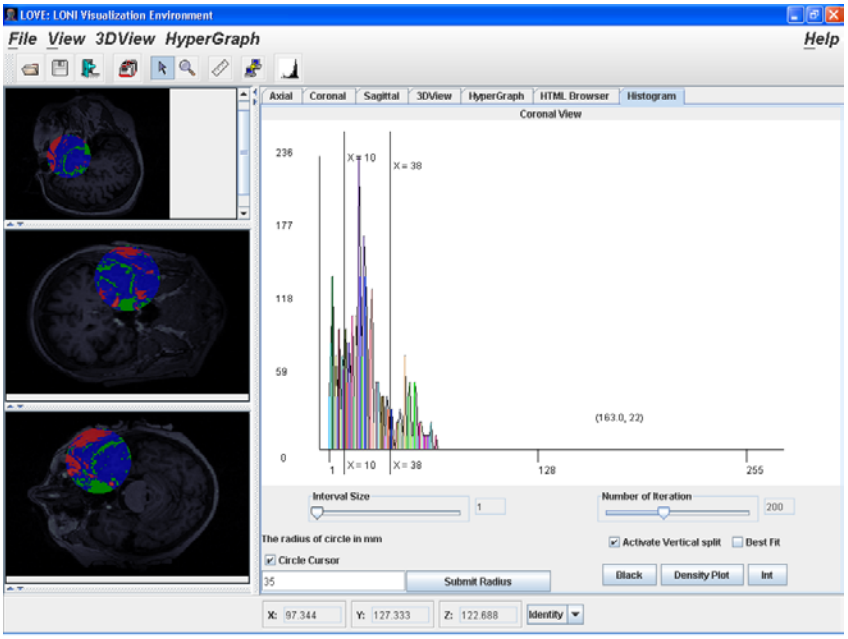

Figure 21. LONI visualization environment showing histogram of selected area of brain.

all three views. The main portion of the GUI shows the histogram of the intensities of the pixels from the selected region. The user can select two points as a starting guess for the intersection points. The algorithm estimates the means from this initial guess and inputs this information along with a starting guess for the variance and weights to the EM algorithm. The EM algorithm runs through a number of iterations and returns the final estimates of the intersection points. The result of the EM algorithm is shown in Figure 22. The estimated normal mixture model is overlaid on the histogram. There is an option in the GUI to visualize the resulting mixture model and the intersection points without the

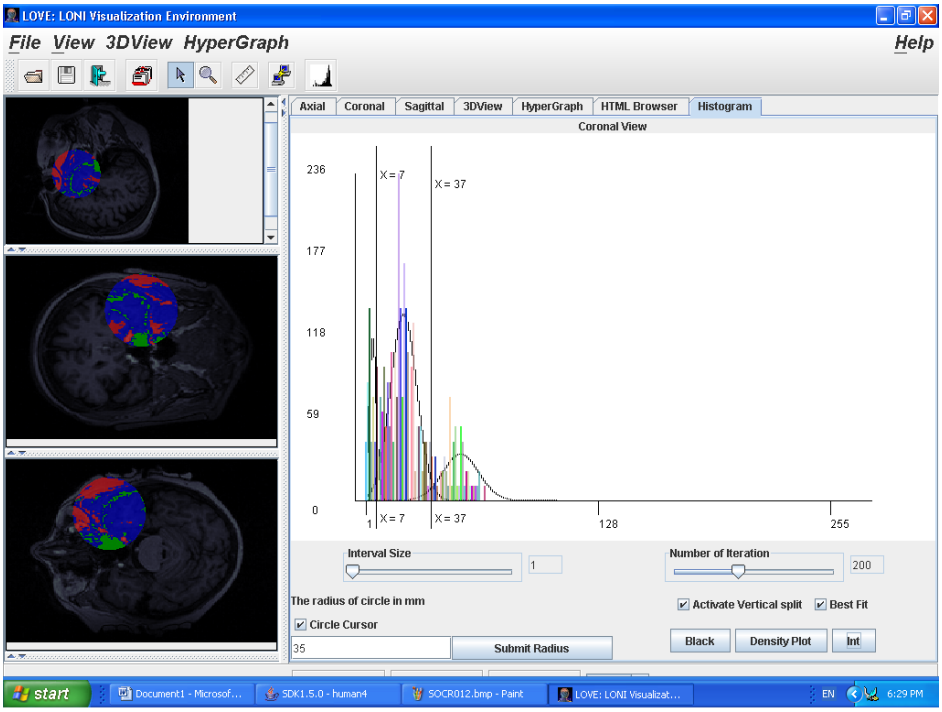

Figure 22. . LONI visualization environment showing mixed model fit superimposed on histogram.

histogram. This is shown in Figure 23. In addition to picking a start point based on user input we randomize some of the initial parameters a number of times and pick the initial starting point that maximizes the log-likelihood. This helps to avoid convergence to a local maximum most of the time.

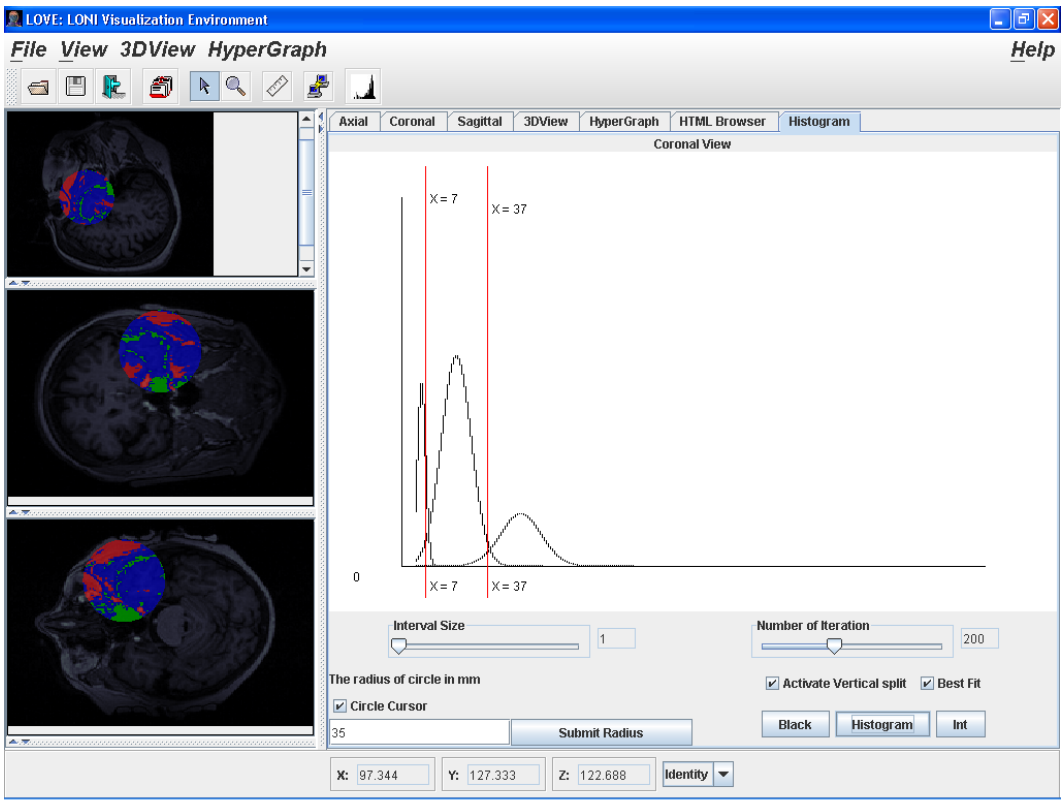

Figure 23. . LONI visualization environment showing model fit with intersection points.

6. FUTURE DIRECTION

The development of SOCR is an ongoing process and is constantly being upgraded and improved upon. There are a number of features that are planned to be added in the near future. The categorical independent variables need to be numerically specified in ANOVA. The ability to use alpha-numeric categories would enhance the usefulness of the application. Currently, ANOVA is capable of handling only balanced data without any missing data. Regression Analysis currently displays results in a tabular format. There is currently a provision for adding plots to the graph panel in the Analysis module. GUI's could be developed for displaying residual plots as part of the regression analysis results.

SOCR Modeler is currently a tool exclusively for modeling probability distribution functions. The next stage in the development of this module would be to integrate basis modelers such as polynomial fits, spline fits, wavelet and Fourier transforms. The framework for wavelet transforms has been completed in the Games section. Currently four wavelets have been implemented. More types of wavelets will be added to the package in the future.

Finally, the SOCR tool would benefit from a study to test its performance. This could be done by testing it's speed, or by comparing it to similar web-based tools or to some user-based benchmarks such as survey results. These additions would help improve the usability of SOCR.

REFERENCES

1. M.H. DeGroot and M.J. Schervish, *Probability and Statistics*, 3rd ed., Addison Wesley, 2001, chap. 6.

2. H.V. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed., Springer-Verlag, 1994, chap. 4, pg. 141 - 204.

3. J.L. Devore, *Probability and Statistics for Engineering and the Sciences*, 5th ed., Duxbury Press, Pacific Grove, CA, 1999, chap. 4, chap. 10, pg. 158 – 169, pg. 402 - 432.

4. H. Stark and J.W. Woods, *Probability and Random Processes with Applications to Signal Processing,* 3rd ed., Prentice Hall, 2001, chap. 2, pg. 60 – 63.

5. J. Bilmes, A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, International Computer Science Institute and Department of Electrical Engineering and Computer Science U.C. Berkeley*, April 1998, ICSI-TR-97-021.

6. T.K. Moon, The Expectation Maximization Algorithm*., IEEE Signal Processing Magazine*, November 1996, volume 13, issue 6, pg. 47 – 60.

7. http://socr.stat.ucla.edu/htmls/SOCR_Distributions.html

8. http://www.stat.ucla.edu/%7Edinov/courses_students.dir/04/Fall/Stat110A.dir/STAT110A_notes.dir/PPCh07.pdf, pg. 1-3,5.

9. J. Neter, M.H. Kutner, C.J. Nachtsheim and W. Wasserman, *Applied Linear Statistical Models*, 4th ed., McGraw-Hill, 1996, chap. 4, 5, pg. 663 – 1034.

10. S.H. Friedberg, A.J. Insel, L.E. Spence, *Linear Algebra*, 3rd ed., Prentice Hall,1997, chap. 6, pg. 315 – 345.

11. A.V. Oppenheim, A.S. Wilsky and S.H. Nawab, *Signals & Systems*, 2nd ed., 1997, chap. 3, pg. 195 – 201.

12. I.D. Dinov, J.W. Boscardin, M.S. Mega, E.L. Sowell,A.W. Toga, A Wavelet-Based Statistical Analysis of fMRI data: I. Motivation and Data Distribution Modeling, *in press*, *NeuroInformatics*, Humana Press, 2005.