University of the Philippines Manila

College of Arts and Sciences

Department of Physical Sciences and Mathematics

Microarray Data Clustering

Using Self-Organizing Maps

A Special Problem In Partial Fulfillment

Of the Requirements for the Degree of

Bachelor of Science in Computer Science

Zach Andrei V. Marasigan

April 2013

ACCEPTANCE SHEET

The Special Problem entitled “Microarray Data Clustering Using Self-Organizing Maps” prepared and submitted by Zach Andrei V. Marasigan in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

____

Geoffrey A. Solano, Ph.D. (candidate)
Adviser

EXAMINERS:

		Approved	**Disapproved**
1.	Gregorio B. Baes, Ph.D. (candidate)	___________	____________
2.	Avegail D. Carpio, M.S.	___________	____________
3.	Richard Bryann L. Chua, Ph.D. (candidate)	___________	____________
4.	Aldrich Colin K. Co, M.S. (candidate)	___________	____________
5.	Perlita E. Gasmen M.S. (candidate)	___________	____________
6.	Ma. Sheila A. Magboo, M.S.	___________	____________
7.	Vincent Peter C. Magboo, M.D., M.S.	___________	____________
8.	Bernie B. Terrado, M.S. (candidate)	___________	____________

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

____

Avegail D. Carpio, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and
Mathematics

____

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

____

Alex C. Gonzaga, Ph.D.
Dean
College of Arts and Sciences

ABSTRACT

Microarray is one of the technologies used in the interdisciplinary science of BioInformatics. Its primary objective is to discover biological knowledge among genes through their expressions. Gene expressions usually come in large and multi-dimensional data which makes computational and statistical analyses necessary. Clustering of microarray data is one of these. Grouping similar genes together unfolds relationships of the biological properties of the genes under specific condition and, if supported by visualization, serves as good decision support for researchers. MaSOM is a software that uses Self-Organizing Maps, an Artificial Neural Network suitable both for clustering and for visualization, as its clustering method. This tool can be used to analyse large data set by preprocessing, clustering, and visualizing two-color cDNA microarray data.

Keywords: Microarray; Gene Expression; Clustering; Self-Organizing Maps

CONTENTS

CHAPTER I: INTRODUCTION

A. Background of the Study

Microarray technology is widely used in Bioinformatics in analyzing thousands of genes. This technology is a recent discovery in molecular biology that requires important but difficult methods in order to arrive to gene expressions that may be helpful in almost every field [1]. Through this, researchers are able to study numbers of genes and samples simultaneously. Gene expression of cells under study can also be observed in genomic level. These gene expressions, which are one of the fundamental concepts in genetics and molecular biology, pertain to the level of protein production by a gene [2]. One of the benefits of using microarray is the detection of cell reactions to drugs or to diseases. Microarray technology thus opens advancement in biological studies such as detection, diagnosis, and treatment. According to [3], this technology enables scientists to monitor complex changes in the expression of genes during cancer development by means of comparing normal and malignant cell samples in the microarray experiments.

The microarray is a small chip containing hundreds of thousands of probes where each of those contains portion of a gene that allows specific gene of the sample to hybridize during the experiment [4]. Its experiment is conducted by extracting RNA from two different cells – one from a normal sample and one from a sample under specific condition under study. The cDNA of both samples, obtained by extraction from their RNA, are then colored with different fluorescent dyes. These dyes are usually colored red and green for the experimental and controlled samples, respectively. After pouring the mixed cDNA to the microarray chip, the genes will hybridize to their complementary. Afterwards, the chip is being cleaned using centrifuge and scanned in a specialized machine to determine the intensities of the fluorescent colors in every spot of the chip. The output data of this experiment is an image of the microarray consisting of spots of different colors. These spots state the expression levels of the gene. To be further used in data analysis, this image undergoes image processing for it to be converted into an array of numerical data. But a major problem in microarray data is the immensity of its dataset wherein not all of its elements are essential in the analysis [5]. Another problem in its scale is the difficulty in characterizing the genes of each sample [2].

Clustering analysis of gene expression data aids in the mentioned problems [6]. Microarray data clustering reduces the dataset into much smaller size containing significant data for better analysis of systematic effects [7]. It is an unsupervised method in grouping data, such as genes, according to a defined similarity criterion [8]. This approach groups genes such that those of the same cluster are similar to each other while different from genes of other clusters.

Self-organizing maps (SOMs) are one of the clustering methods that can be used in microarray data. This is very appropriate in data analysis because of its ability to cluster data in reduced dimensions and its easy configuration to visualize the data [9-13]. This neural network, which clusters data based on the concept of neighboring association, was proven to be efficient in visualizing thousands of data by the use of small number of data points as its representation [11]. Furthermore, the use of self-organizing maps is found to be robust, accurate, easy to implement, reasonably fast, and scalable to large data sets [9].

In this study, self-organizing maps is used to cluster microarray data in order to discover and visualize patterns in gene expressions.

B. Statement of the Problem

The dimension of microarray data is very large and this makes gene expression analysis a complex process. It is beneficial for researchers to make use of a tool that helps them to automatically preprocess and cluster gene expressions. Moreover, visualization is also desirable for these large data sets during exploratory analysis and as a decision support tool for the researchers. Hence Self-Organizing Maps, perceived to be a promising solution due to its suitability in both clustering and visualization, is used to cluster microarray data.

C. Objectives of the Study

Generally, this study aims to solve the problems presented by providing a tool that allows researchers to automatically preprocess and cluster inputted microarray data and obtain a visualization of its results. Specifically, these can be done by providing the following functionalities of the software to any user interested in studying microarray data.

a. Cluster data by means of Self-Organizing Maps

b. View the heatmap representation of the clustered data

c. Preprocess microarray data through any of the available methods:

    i. Logarithmic Transformation

    ii. Z-Score Transformation

    iii. Median Centering

    iv. Scale Normalization

    v. Quantile Normalization

d. Import a file containing the microarray data

e. Optionally import files containing descriptions of the microarray data to be used

f. Optionally set values to the following parameters of the clustering method

    i. Dimension of the Self-Organizing Map

    ii. Maximum number of iterations for the clustering

    iii. Initial learning rate of the training

    iv. Distance measures such as

        1. Euclidean metric

        2. Manhattan metric

        3. Sup-norm metric

g. Export the results into:

    i. Portable Document Format file (.pdf)

    ii. Spreadsheet (.xls)

    iii. Image (.png)

    iv. Text file (.txt)

h. View the user's guide on how to operate the tool

D. Significance of the Study

Microarray technology relies on useful statistical methods and can be enhanced through data visualization. Presentation of clustered data is very essential in exploratory data analysis especially for the practitioners because it gives them clearer view of the relationships between the divisions and eases their interpretation of the results [14].

According to [15], analysis of significantly large data such as gene expressions brings out statistical and computational problems to biologists when not solved using the new and continuously growing statistical techniques. Furthermore, clustering large data gives researchers initial knowledge about the nature of the data and thus serves as a decision support for their actual experimentation.

Thus, there is a need to implement a solution that is well suited in both gene expression analysis and its presentation. This study of implementing Self-Organizing Maps as a clustering and visualization method of microarray data is essential in simplifying and generating visualization of results in microarray experiments for better analysis of medical or statistical practitioners. It is beneficial to provide the users an application that allows them to automatically preprocess microarray data, cluster those using Self-Organizing Maps, and observe the generated visualization of the results of the analysis.

E. Scope and Limitations

The primary goal of this study is to provide researchers a tool that clusters microarray data. However, along with the study's objectives are the following limitations:

a. The system accepts numerical representation of microarray data as its input.

b. The data input is in .txt file extension and follows the format specified in the methodology.

c. The user may provide at most one description file for genes and one description file for samples.

d. The clustering of the Self-Organizing Map relies on parameters or factors set by the user or the system.

e. The output clusters may vary in different iterations due to the randomization in the techniques and to the different effects of the parameters used.

f. The validation and further analyses of results are tasks of the user since the system only serves as a supporting tool for the actual research.

g. This study does not intend to provide a clustering tool that is best applicable for all types of data.

F. Assumptions

The following are assumed throughout the implementation of the system design:

a. The data to be processed by the tool are two-color cDNA microarray data.

b. The system assumes that the data input have good quality, i.e., no missing fields, no corrupted parts, and are consistent in their dimension and format.

c. The end users are assumed to be knowledgeable of the parameters, input variables, and the underlying concepts of microarray and its clustering.

CHAPTER II: REVIEW OF RELATED LITERATURE

It is generally acknowledged in most literatures that microarray has been useful in simultaneous analysis of large datasets of gene expressions. This technology provides the possibility of detecting gene variations under the influence of chemicals, drugs, or diseases that enables experts to detect and study various conditions of the organism of interest.

Many researches are being conducted in improving applications of microarrays to study diseases. Hewett and Kijsanayothin in [3] used classification rankings in microarrays in tumors categorization. Also, Zhang, Ding, and Li in [2] used reliefF and mRMR methods in interpreting the microarray data of breast cancer, several types of leukemia, and various tumors such as prostate, lung, colorectal, lymphoma, bladder, melanoma, uterus, leukemia, renal, breast, pancreas, ovary, mesothelioma, CNS, and MET.

Venkatesh and Thangaraj in [5] classified gene expression of biopsy samples collected from colon cancer patients with the use of Linear Vector Quantization. A Rough Set based Gene Expression Clustering Algorithm was used in analyzing colon cancer dataset in [6] by Emilyn and Ramar. And in [4], Bhuvaneswari and Brintha used a type 2 fuzzy logic approach in converting microarray data into fuzzy terms. Fuzzy association patterns are then used to cluster the data. They also mentioned that neural networks can be incorporated in this methodology.

Also, many methodologies and approaches are being developed in trying to solve the challenges of microarray technology and to extract the most significant result in the most efficient manner. Richard et al in [7] found out that K-means clustering, considered alone, was probably better than any compared method in the study, namely, CRC, ISA and memISA. With the use of three brain expression datasets, they had found out that K-means is relatively fast, does not rely on parallelization, and clusters data with richer gene ontology, but they claimed that these methods can be combined to improve their performance. But in [8], Lani and Rajalakshmi claimed that K-means has disadvantages such as its dependency on the initial parameter k which pertains to the user's desired number of clusters. Also, resulting clusters in K-means method differ on every repetition since the quality of its clusters depends on the method's randomly chosen centroids. This problem had been solved by the proposed Quad tree based EM algorithm that gave more compact clusters based on their assessment through Silhouette coefficient.

Various clustering algorithms were compared by Sharma and Rungta in [1]. The most popular algorithms namely Hierarchical, K-means, Fast Genetic k-Means, Incremental Genetic K-Means, Fuzzy C-Means, Expectation Maximization, neural network-based (Self-Organizing Maps), and density-based (Density-Based Spatial Cluster of Applications with Noise) clustering methods were compared in terms of data uncertainty, optimization, and density estimation. The datasets used in the comparative study were leukemia, colon, and lymphoma gene samples. They claimed that there is no perfect universal solution and found out that there is no absolute best or perfect algorithm among these.

Scharl, Voglhuber, and Leisch in [14] introduced an interactive visualization tool gcExplorer to clarify the very complex relationship between clusters of gene expressions. They emphasized the importance of data visualization of clustered microarray data for the benefit of the practitioners. In line with this idea, McGarry, Sarfraz, and MacIntyre in [13] used SOM as microarray data clustering method because of its dimension reduction, clustering, and visualization capabilities. The objective of their study was to let SOM discover significant groups of genes that were present in a particular cancer type. They also aided this method with U-matrix technique and Gene Ontology for provide further analysis of the clustered data results and for extraction of biologically meaningful observations.

Vanichayobon et al in [10] proposed an algorithm using statistical methods and Self-Organizing Maps in clustering and rule-creation in microarray data. This technique arrived simple and few rules after the gene selection. They had used carcinoma, leukemia, and lung cancer as their datasets and arrived at cancer prediction rules of 100% accuracy. Their experiment claimed that the proposed algorithm can be further applied to other diseases being tested in microarrays. Also in [9], Tamayo et al used a computer package called GENECLUSTER together with Self-Organizing Maps in gene expression analysis. They found that SOM is well suited in data exploration and classification of multidimensional data such as microarrays in order to arrive at essential data insights.

Yi and Hasan in [11] studied the modified Self-Organizing Maps method that was combined with fuzzy c-means rules. They used this method to cluster and present leukemia and brain tumor microarray datasets and claimed that the interpretation of results of SOM method is dependent on the familiarity of the user and produces less desired results than the modified method. But Torkkola et al in [12] mentioned that clustering using displays may be a disadvantage in deterministic purposes but considered an advantage in interactive exploratory analysis. They stated that SOM does not require the data to undergo a critical preprocessing technique of missing value filling. In their study, Self-Organizing Maps was used in exploratory analysis of yeast DNA data in order to find patterns very rapidly and to find new genes of similar characteristics of expression and possibly new families of genes.

CHAPTER III: THEORETICAL FRAMEWORK

A. Gene Expression

Microarray data is represented as (m x n) array, called gene expression matrix, with which the n columns are labeled as the samples and each of the m rows represent the genes [16]. The collective column data are called the gene expressions of the corresponding sample. Each grid contains numerical data that were obtained by processing the microarray image. Each of these numerical data, called expression ratio, is a relative measurement between the intensities of red and green colors in its corresponding spots in the image [17]. The formula below gives the ratio of the color intensities R and G for red and green, respectively.

$$ratio = \frac{R}{G}$$

Equation 1: Expression Ratio

B. Microarray Clustering

Grouping microarray data may be based on either of the two possible ways [18]. One of which is clustering of samples based on their gene expression profiles. In this manner, we group the m columns of samples based on the n variables of genes. Clustering samples of similar expression levels may be helpful in possibly identifying subclasses or taxonomy of the disease or condition of interest. The other clustering approach is grouping m genes according to their similarity in the n samples. This can illustrate the behaviors of genes if the samples are spatially or temporally expressed. As both orientations of clustering are beneficial, several methods group the microarray data in both manners.

C. Self-Organizing Maps

Self Organizing Maps (SOMs), a type of Artificial Neural Network, is also called Kohonen Self-Organizing Maps, named after its Finnish inventor, Professor Tuevo Kohonen [19]. It is an unsupervised method of competitive learning that is specialized in data dimension reduction and visualization without altering the topology, or relationships, between the data elements. In this neural network, the large dimension of data of interest is being visualized into usually one- or two- dimensions such that similar data elements are grouped together and dissimilar elements are at different groups.

SOM uses a specified dimension of map composed of weighted nodes known as neurons. These nodes are dynamically adjusted to conform to the structure of the input data. Each of the neurons is associated with a weight vector of similar dimension as the vector of the input data. As an example, shown in the succeeding figure is a three-by-four map whose nodes have a weight vector of size three.

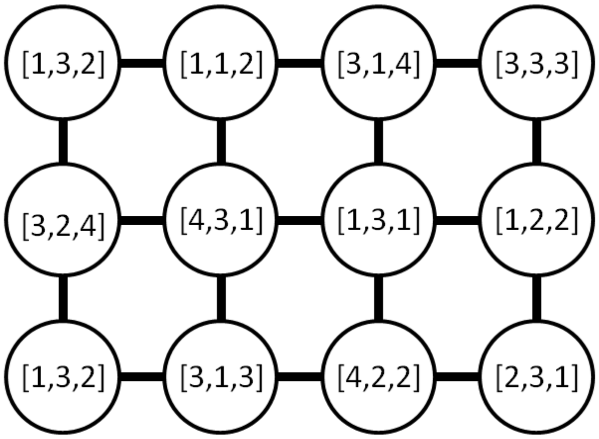


Figure 1: Sample map of weighted neurons

Input data set is clustered by iteratively adjusting the map until it finally arranged its nodes based on the natural grouping of the elements or when the provided maximum number of iterations is reached. Initially, this method sets random weights to all of the nodes of its map. One by one, it inserts input data vectors and looks for its Best Matching Unit (BMU) or the closest node among all the neurons of the map. Then, the weight vectors of all of the neighbors of the BMU will be modified based on the formula

$$\mathbb{W}_n(t+1) = \mathbb{W}_n(t) + \Theta(t)\, L(t) \left[\mathbb{D}_i(t) - \mathbb{W}_n(t)\right]$$

Equation 2: SOM Learning Function

where t represents the time or iteration, $W_n(t)$ and $W_n(t+1)$ are the next and previous weight vectors, respectively, of the neighboring node n, and $D(t)$ is the vector of the input data. $\Theta(t)$ is called the neighborhood function, a formula that determines the closeness of the BMU to other nodes of the map which is given by

$$\Theta(t) = e^{-\left(\frac{d^2}{2\sigma^2 t}\right)}$$

Equation 3: Neighborhood Function

where d is the distance between the BMU and its neighbor and σ is the radius of the neighborhood function obtained by Equation 4.

$$\sigma(t) = \sigma_0 e^{\left(-\frac{t}{\lambda}\right)}$$

Equation 4: Neighborhood Decay

Here, σ_0 is the starting radius of the map and λ is the time constant. The latter is equivalent to the maximum number of iterations divided by the logarithm of the starting radius.

The $L(t)$ in SOM's learning function is called the learning rate whose value typically starts at one and ends with nearly zero after the clustering. This function is dependent on the initial learning rate L_0 and the maximum number of iterations denoted by λ. It determines the amount of influence of the input vector to the new weight of the neighbor. It can be obtained by using the formula

$$L(t) = \mathrm{L}_0 e^{\left(-\frac{t}{\lambda}\right)}$$

Equation 5: Learning Rate

Formulas for Self-Organizing Maps presented above were obtained from [19]. Note that in this system, the steps in clustering methods are based on these functions and that clustering data through SOM can be performed in several variations.

D. K-Means

K-Means is a method that is widely used in clustering data. In this method, data points are clustered into k groups. As it partitions the data, each point selects the nearest of the k centroids. Then, this closest centroid adjusts its weight so that it will become the midpoint of its members including the new one. This process repeats until all the centroids have stopped adjusting [7]. Although this technique is considered to be effective, the number of resulting clusters highly depends on the user input. Thus, the user must have prior knowledge onto what must be the number of clusters to come out from the given data.

E. Distance Measures

Most clustering methods use dissimilarity measures to determine the relationships between the vectors to be grouped. The commonly used dissimilarity measures in clustering are Euclidean, Manhattan, and Sup-norm distance formulas and are described, respectively, by the succeeding formulas.

$$d(a, b) = \sqrt{\sum_{i=1}^{n} \left(a_i - b_i\right)^2}$$

Equation 6: Euclidean Distance

$$d(a, b) = \sum_{i=1}^{n} \left| a_i - b_i \right|$$

Equation 7: Manhattan Distance

$$d(a, b) = \max_{1 \leq i \leq n} \left| a_i - b_i \right|$$

Equation 8: Sup-norm Distance

In these methods, a_i is the i^{th} gene expression ratio of sample a and b_i is the i^{th} gene expression ratio of sample b [17]. The output *d* is called the distance, or the difference, between points a and b. If the two points are equal, distance *d* is equal to zero. Otherwise, *d* is a non-negative value.

F. Preprocessing

Microarray data usually undergo several preparations before the actual analysis. These processes are being performed in order to produce more satisfactory results. Due to possible variations in the microarray experiment, it is essential to perform normalization on the microarray data to ensure reasonable comparison between them [13]. The technical differences, experimental biases, and non-biological variations in color intensities in each experiment are being reduced. This technique, which is called data normalization, is essential to be performed among observations of expression levels.

One of the methods used is constant normalization or commonly known as the z-score transformation. In this process, the mean among the variables is set to 0 and the variance among them is set to 1. This is done by using the formula below [18] to guarantee that the data are distributed normally and to eliminate the possible differences and experimental biases brought by the external factors. The mean $E(x)$ is given by

$$E(x) = \frac{\sum_{i=1}^{n} x_i}{n}$$

Equation 9: Mean

where the x_i's are the data points of the vector or, for instance, gene expression ratio of a given sample. The standard deviation s*(x)* can be obtained by using

$$s(x) = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - E(x)\right)^2}{n-1}}$$

Equation 10: Standard Deviation

So that each of the expression ratios x_i of the sample can be transformed into its new value computed using then next formula.

$$\frac{x_i - E(x)}{s(x)}$$

Equation 11: Z-Score Transformation

Logarithmic transformation is another approach to normalization. This method is performed to the data especially when the points extremely vary from the other. This process can be done by using the formula

$$\log_2 \left(x_i\right) = \log_2 \left(\frac{R}{G}\right) = \log_2 (R) - \log_2 (G)$$

Equation 12: Logarithmic Transformation

With the use of logarithmic transformation, the continuity of the data points is ensured [17]. These logarithmic expression ratios signify that points equal to zero have equal intensities of red and green colors. Also, the more red color dominates the spot, the more positive its gene expression becomes. Similarly, as the value becomes more negative, it signifies that the spot becomes greener. Thus, logarithmic transformation eases the interpretation of the numerical representation of microarray images.

Median Centering is simply the process of subtracting each data points by the median of the group which they belong to. Median is a measure of central tendency which is defined as the point located at the middle of all the members of the collection of data. By subtracting it to each of the data points, the median of the collection will become zero and its members will be divided primarily into two groups – one of which consists of positive numbers or the points above the median and the other which consists of negative numbers or those which are located below the median.

Scale Normalization is a preprocessing technique that is applied to expression matrix that has been logarithmic transformed and median centered. As proposed by [20], this method enforces the data samples to have equal variances by assuming that each group i have mean equal to zero and variance equal to $a_i^2 \sigma^2$. It will then transform the data so that the factor a_i^2 will be scaled to one, leaving the samples with equal variances of σ^2. This process, as described by the succeeding equations 13 to 15, adjusts the a_i^2 of each sample i which is dependent on median-absolute-deviations or the medians of the median-centered ratios. This can be done by multiplying all the data of each sample with the scale which is equal to a_i divided by its median-absolute-deviation.

$$a_i = \frac{MAD_i}{\sqrt[n]{MAD_i}}$$

Equation 13: Scale of sample *i* with respect to the common variance

$$MAD_i = median(|x_{1,i} - M_i|, |x_{2,i} - M_i|, ..., |x_{m,i} - M_i|)$$

Equation 14: Median-Absolute-Deviation of sample *i*

$$M_i = median(x_{1,j}, x_{2,j}, ..., x_{m,j})$$

Equation 15: Median of sample *i*

Last of the preprocessing methods is the Quantile Normalization which forces the distribution of every data sample to become similar. This can be done by averaging the data points among all the groups belonging to the same quantile and then replacing those values with the computed mean. With this, the distribution of all the data samples becomes straight or, more specifically, becomes empirical distributions which have identical quantiles. Its steps include sorting the data of each of the samples, averaging data points that belong to the same ranking, replacing the value of those data with the computed average, and then reordering all the samples back into its original order when all points have been processed [21].

G. Heatmaps

Heatmaps are one of the visualization methods used in microarray data. In this approach, a range of colors also known as shade is used to denote variation in the dimension of data points. Heatmaps provide a larger view of the gene expression levels of the samples by displaying different intensities. The following figure is an example of a heatmap of a hierarchically clustered gene expression obtained from [22]. In this example of heatmap, green color represents lowly expressed genes and red or black colors represent higher expressed genes [18]. With the aid of varying color intensities relative differences in the very large dataset can easily be observed especially when the data are already grouped together.

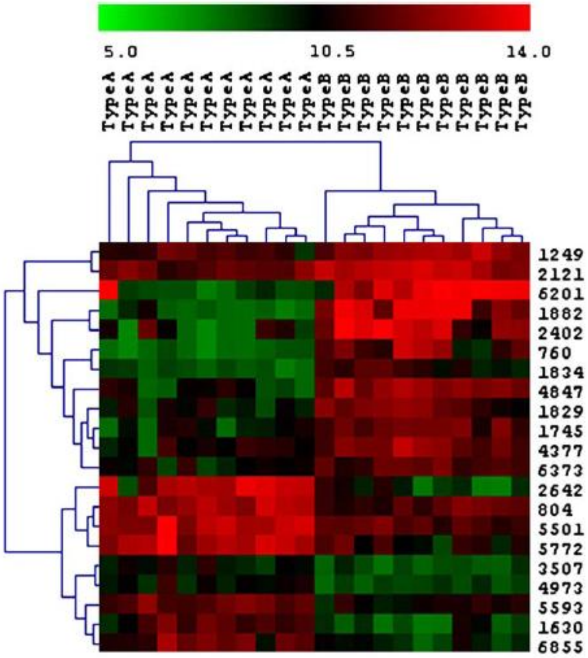


Figure 2: Sample heatmap

CHAPTER IV: DESIGN AND IMPLEMENTATION

A. Data Specifications

The user inputs the microarray’s gene expression matrix by providing a file that strictly follows the format specified below:

a. Its format is .txt.

b. It is composed of columns and rows of data that are tab-delimited.

c. Its columns correspond to the experimental samples.

d. Its rows correspond to the genes.

e. Its first row contains numeric or alphanumeric data that serves as the identification key of the sample at the corresponding column.

f. Its first column contains numeric or alphanumeric data that serves as the identification key of the gene at the corresponding row.

g. It does not contain any missing data.

The user can optionally submit an additional file that describes the samples or the genes used in the given microarray data. It is expected to be in the format described below:

a. Its format is .txt.
b. It is composed of columns and rows of data that are tab-delimited.
c. Its first row contains the header of the descriptions of the samples or genes.
d. Its first column consists of the numeric or alphanumeric identification key of the samples or genes in the same order as how they are listed in the file for the gene expression data.
e. Details do not contain any tab characters.
f. Details may be missing but the first column, or the identification keys, must be completely listed.

There are many freely-available sample data in the internet. Two of the recommended websites that offer open access to processed microarray data are Gene Expression Omnibus in http://www.ncbi.nlm.nih.gov/sites/GDSbrowser and BRB-Array Tools for Human Cancer Gene Expression in http://linus.nci.nih.gov/~brb/DataArchive_New.html [23]. But the available files from those databases require few modifications in order to completely conform to the format specified in the system.

B. System Design

The application is implemented as defined in the succeeding diagrams. Generally, the tool's input and output requirements are defined in the context diagram below.

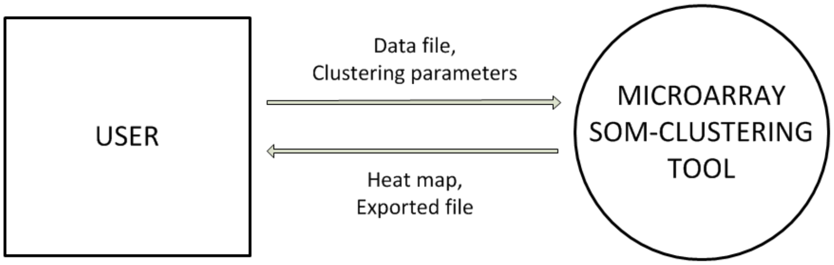


Figure 3: Context Diagram, MaSOM

The system gathers the microarray data from the user through the provided data file. The user can also specify clustering parameters, namely, the desired number of clusters, the maximum number of iterations, and the distance measure. If the user has not specified any, the system uses the default values for the mentioned variables. As a rule of thumb used by statisticians in clustering, the default number of cluster is 4 and the maximum number of iterations is five times the number of data to be clustered. The default distance metric to be used is the Euclidean distance since it is the most commonly used and familiar formula of dissimilarity in clustering. The user has the option to submit files that describe the microarray data to add supplementary details for the user's interpretation of results. Lastly, the system also gives the user the capability of exporting the results in PDF, text, spreadsheet, or image files depending on what details are needed. An elaboration of the system's invoice is provided by this use-case diagram.

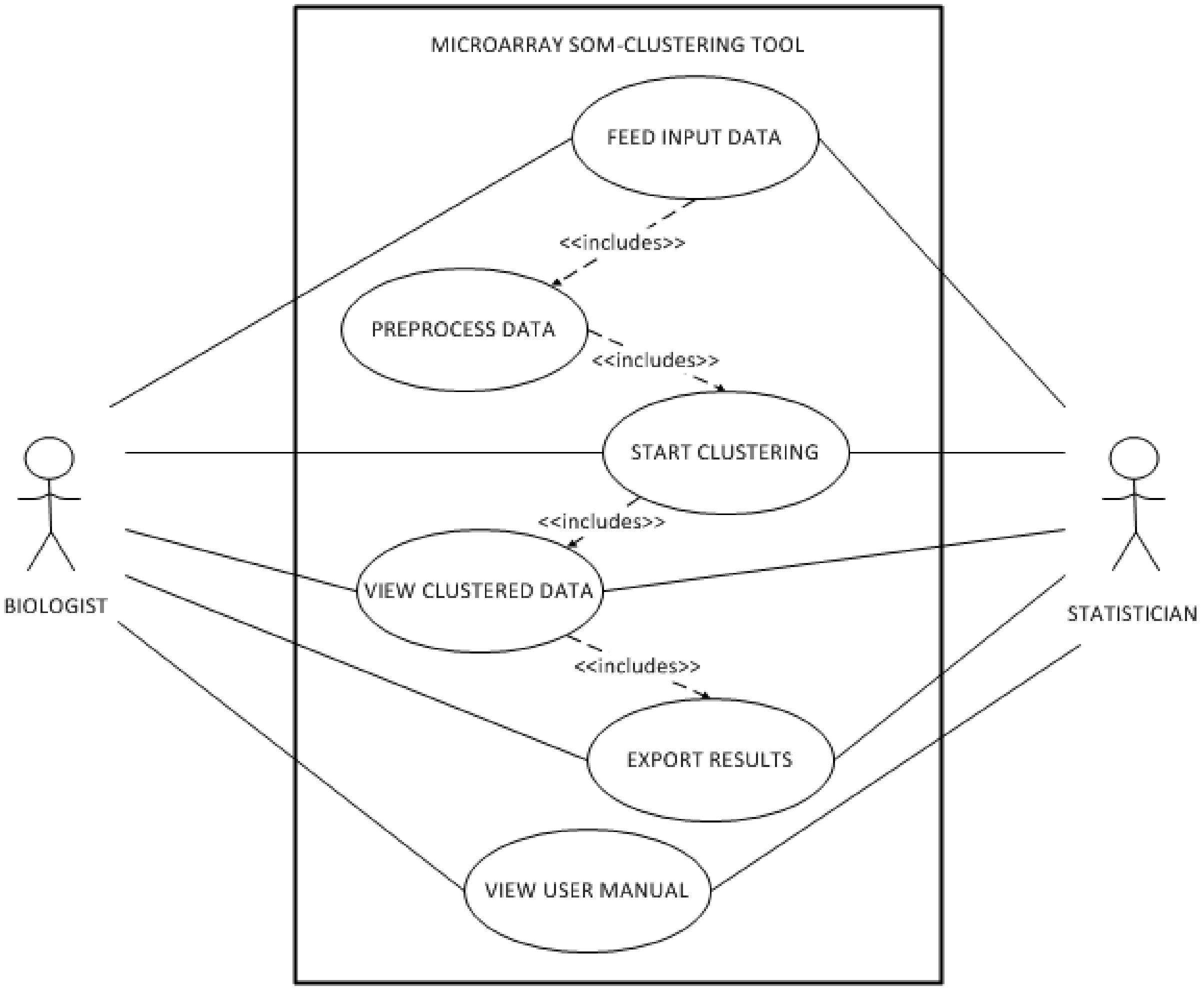


Figure 4: Use-Case Diagram, MaSOM

Basically, the clustering tool aids the users by providing better visualization of the multi-dimensional gene expression levels. Since it is intended to offer this system to anyone interested in analyzing microarray data, some users may not be familiar on how the tool is being used. Thus it is necessary to provide user manual containing instructions and pointers on how to use the tool properly.

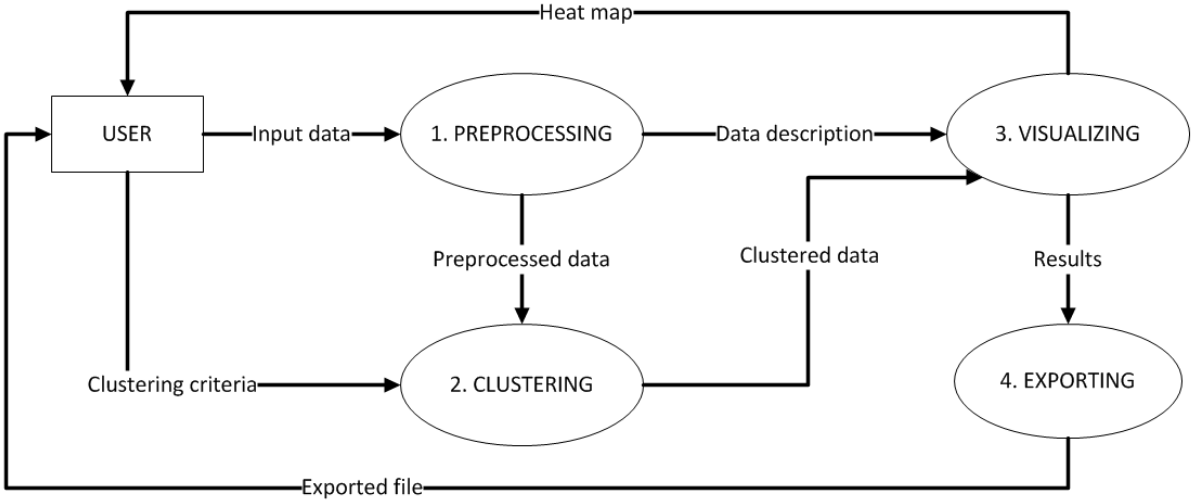


Figure 5: Data Flow Diagram, MaSOM

The functionalities of the system are defined in the above data flow diagram. The tool, after receiving the input data from the user, first preprocesses the data. As shown in the sub-explosion of this process in Figure 6, the data are extracted from the file through parsing. These data may undergo various preprocessing methods to eliminate possible experimental biases and to make comparisons between data points reasonable. In order for the users to come up with more desirable results, they are given the option to choose from the methods provided by the system depending on how they want the data to be preprocessed.

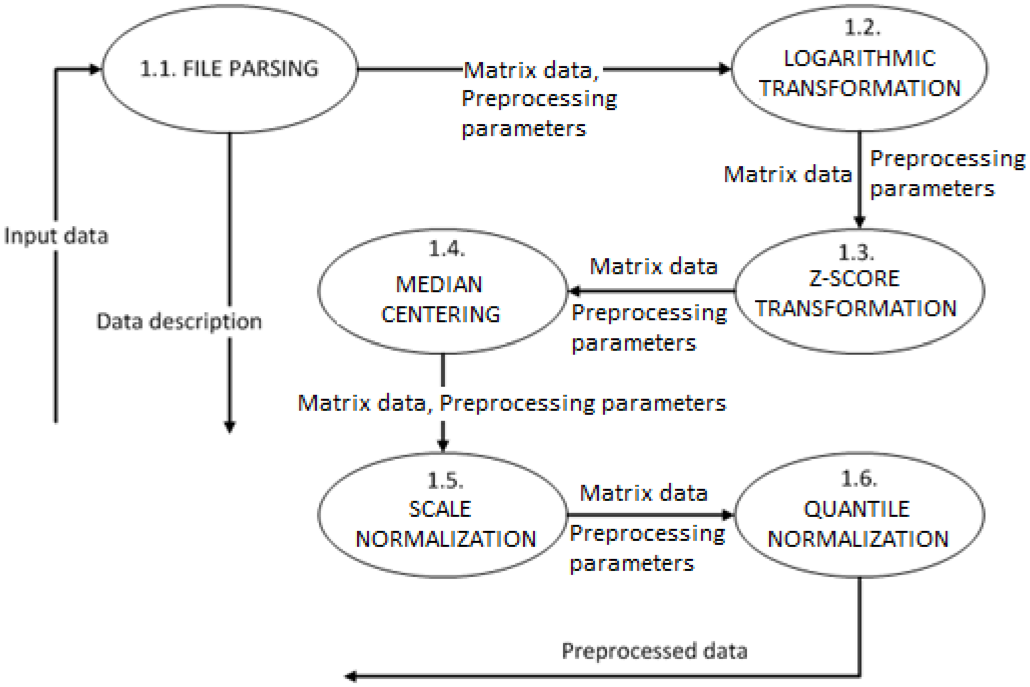


Figure 6: Sub-explosion of Process 1, MaSOM

Right after preprocessed data are obtained, the Self-Organizing Map, as shown in Figure 7, prepares its functions and data structures during Process 2.1. It then undergoes successive training based on the data being fed by the system. After it reaches the convergence criterion, K-means clustering finalizes the groups.

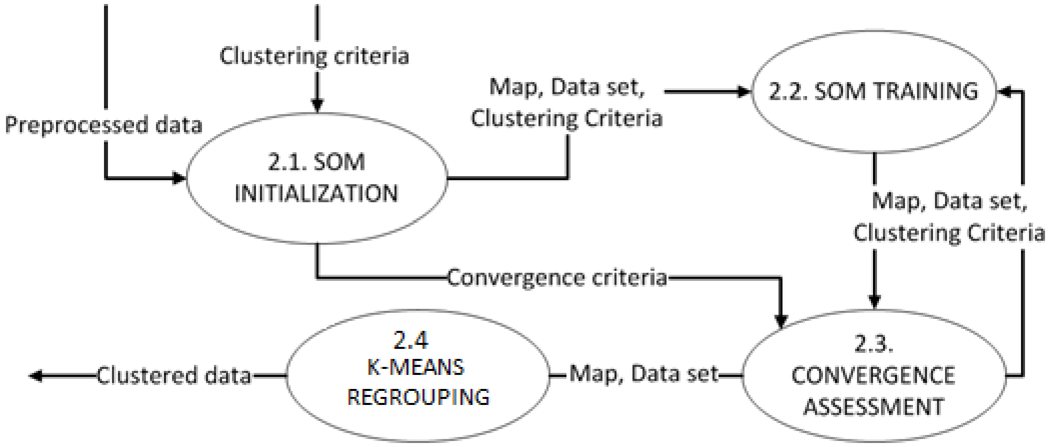


Figure 7: Sub-explosion of Process 2, MaSOM

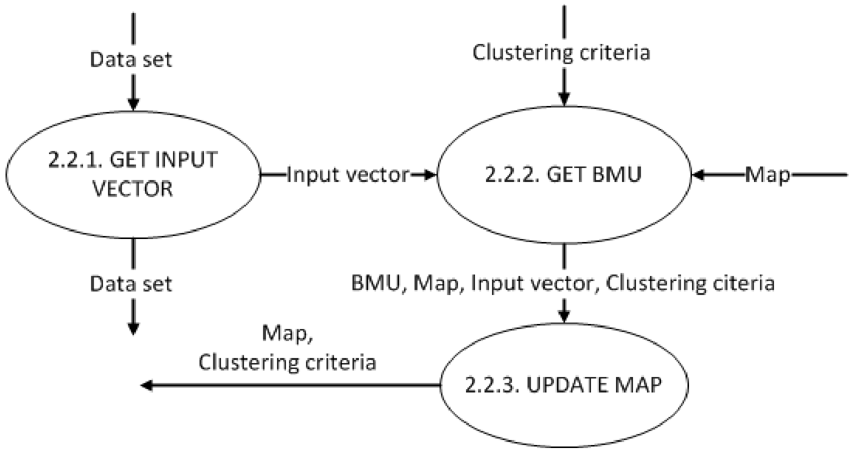


Figure 8: Sub-explosion of Process 2.2, MaSOM

The diagram above is the sub-explosion of the SOM training process. It is illustrated that each input vector, the gene or sample vector, from the data set are introduced to the map nodes. During this phase, the algorithm searches for the best matching unit (BMU) which is defined as the node closest to the input vector. The BMU is obtained through the process described in Figure 9.

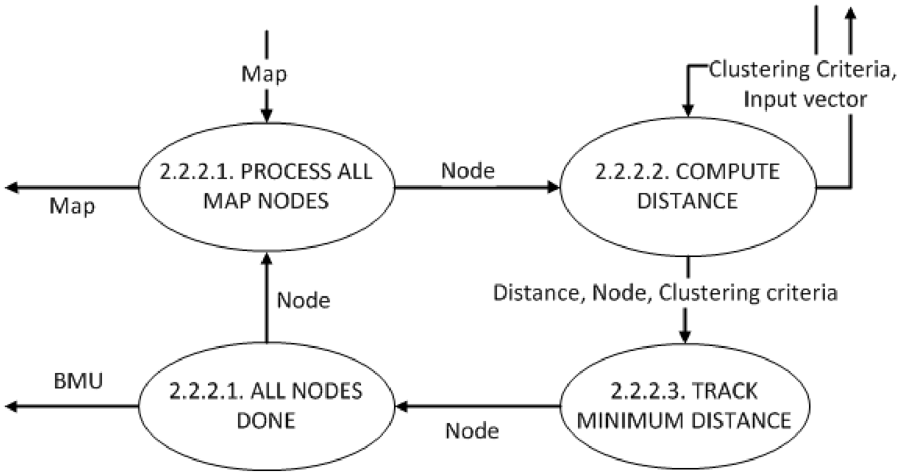
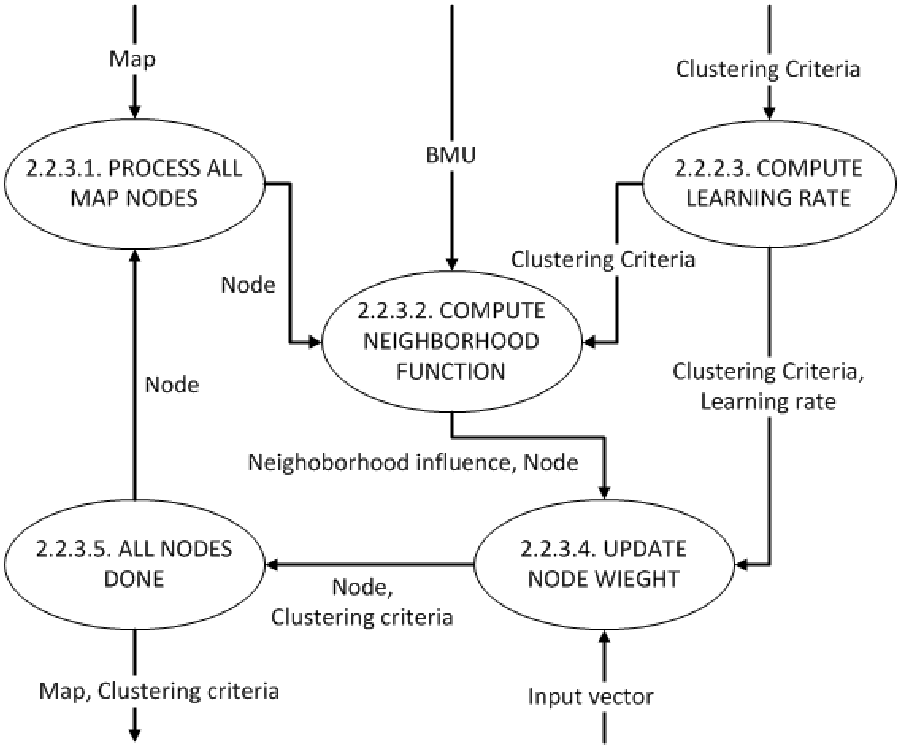

Map
Clustering Criteria,
Input vector
2.2.2.1. PROCESS ALL MAP NODES
Node
2.2.2.2. COMPUTE DISTANCE
Map
Node
Distance, Node, Clustering criteria
BMU
2.2.2.1. ALL NODES DONE
2.2.2.3. TRACK MINIMUM DISTANCE
Node

Figure 9: Sub-explosion of Process 2.2.2, MaSOM

In this process, the distance between the input vector and all the nodes of the map are computed using the selected distance metric. As a new distance is obtained, the algorithm keeps track of the node nearest to the input vector. The winning node, now called the BMU, is then passed to the process of updating the map detailed in the next diagram.



Map
BMU
Clustering Criteria
2.2.3.1. PROCESS ALL MAP NODES
2.2.2.3. COMPUTE LEARNING RATE
Clustering Criteria
Node
2.2.3.2. COMPUTE NEIGHBORHOOD FUNCTION
Node
Clustering Criteria, Learning rate
Neighoborhood influence, Node
2.2.3.5. ALL NODES DONE
2.2.3.4. UPDATE NODE WIEGHT
Node, Clustering criteria
Map, Clustering criteria
Input vector

Figure 10: Sub-explosion of Process 2.2.3, MaSOM

The Self-Organizing Map modifies the geometry of its nodes by using its neighborhood function, learning rate, the values of the input vector, and the BMU's weight. In this phase, the neighboring nodes influence the shifts of the processed node based on the weight obtained from the neighborhood function. After all the nodes are updated, the algorithm then proceeds to check for the convergence of the clustering process while referring to the maximum iteration or on the immovability of the map.

The system provides visualization of the clustered data using heatmaps that represents the expression levels of the genes. If descriptions of the samples or genes are provided, the tool provides associating details along with the clusters. Finally, if the user opts to, the system exports its output into chosen files for further use of the researchers or for archiving.

As an elaboration of the methods presented in the diagrams, algorithms of the primary functions are enumerated. Note that these algorithms are in pseudocode and do not conform to syntax of any programming language.

Algorithm 1: Self-Organizing Maps

```
for all nodes i in SOM
    for all items j of i’s weight W[i]
        // randomize weight vectors of all nodes
        W[i][j] = Math.random();
while t < maxIter
    for all input vectors D[] in the dataset
        W[BMU] = infinity;
        for all nodes i in SOM
            // search for Best Matching Unit
            if dist(D[k],W[i][],vector_size) <
               dist(D[],W[BMU][],vector_size)
            then BMU = i;
        for all nodes i in SOM
            // adjust neighboring nodes
            W[i][] = W[i][] + neighborhood(t,BMU,i) *
            learning_rate(t) * ( D[] - W[i][]);
        t++;
```

Algorithm 2: Heatmap

```
//determine the range of values
max_value; min_value;
for all sample vectors D[s] in D[m][n]
    for all gene expressions g in D[s]
            max_value = Math.max(D[g][s], max_value);
        min_value = Math.min(D[g][s], min_value);
range = max_value - min_value;
C[m][n]; // array of colors
// assign a color based in value
for all sample vectors color C[s] in C[m][n]
    for all gene expressions color g in C[s]
        double norm = (D[m][n] - min_value) / range; // 0 ≤ norm ≤ 1
        int colorIndex = Math.floor(norm * (colors.length - 1));
        C[m][n] = colorIndex;
// display heatmap
display(sample_id);
for all gene expressions color g in C[g] in C[m][n]
    display(gene_id);
    for all sample vectors color C[g]
        display(C[m][n]);
```

Algorithm 3: Manhattan Distance

```
float Manhattan_Dist ( A[], B[], vector_size )
      for i from 0 to vector_size - 1
            dist += Math.abs( A[i] – B[i] );
      return dist;
```

Algorithm 4: Euclidean Distance

```
float Euclidean_Dist ( A[], B[], vector_size )
      for i from 0 to vector_size - 1
            dist += Math.square( A[i] – B[i] );
      return Math.sqrt(dist);
```

Algorithm 5: SupNorm Distance

```
float SupNorm_Dist ( A[], B[], vector_size )
      dist = 0;
      for i from 0 to vector_size - 1
            dist = Math.max(Math.abs( A[i] – B[i] ),dist);
      return dist;
```

Algorithm6: Logarithmic Transformation

```
for all sample vectors D[s] in D[m][n]
      for all gene expressions g in D[s]
            // $x = log_2(x)$
            D[g][s] = Math.log(D[g][s], base=2);
```

Algorithm 7: Z-Score Transformation

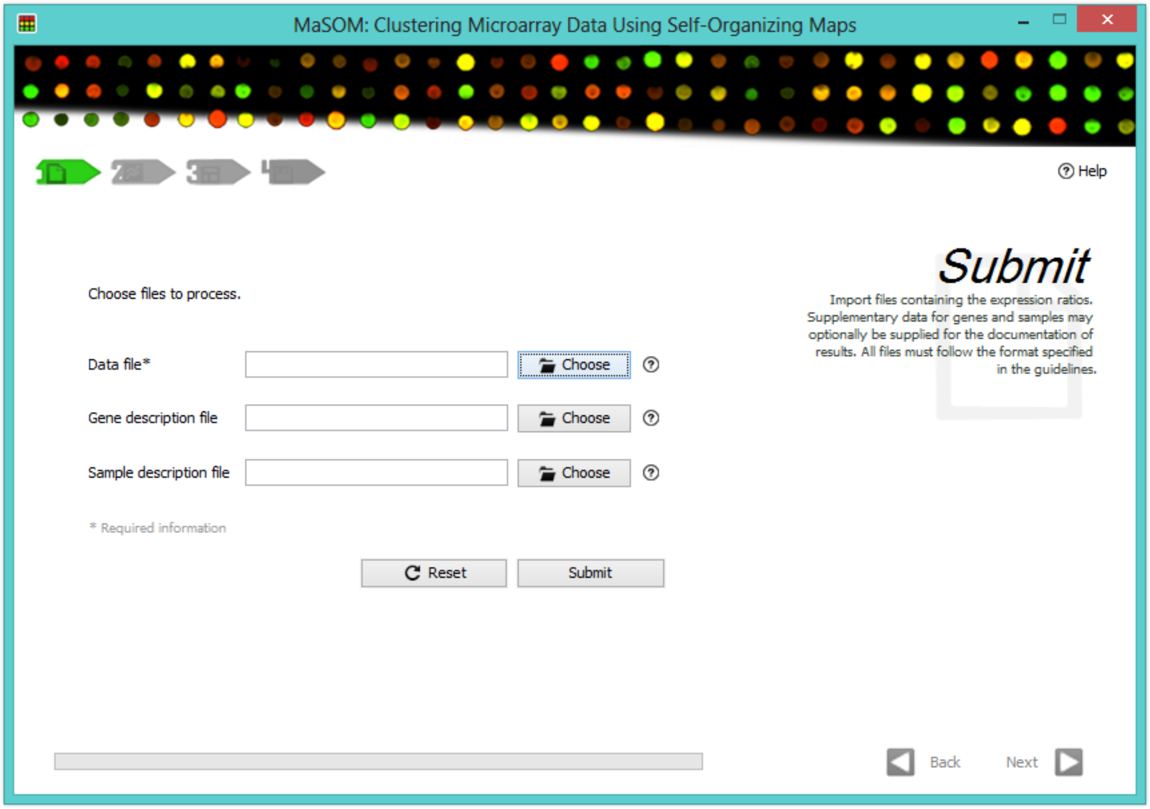
```
for all sample vectors D[s] in D[m][n]
      sumG = 0, sdG = 0, sdG2 = 0;
      for all gene expressions g in D[s]
            sumG += D[g][s]; // summation of x’s
      meanG = sumG / m; // mean of x or E(x)
      for all gene expressions g in D[s]
            // summation of $(x – meanX)^2$
            sdG += Math.square( D[g][s] - meanG );
      sdG = Math.sqrt(varG2) / (m-1) // standard deviation of x
      for all gene expressions g in D[s]
            // actual normalization
            D[g][s] += (D[g][s] - meanG ) / sdG;
```

C. Technical Requirements

The system is implemented in Java programming language. Thus a machine installed with Java Runtime Environment is required to run the tool. Also, since microarray dataset is significantly large and that the tool uses computations that are time- and memory space-expensive, it is advisable to use the system in a machine with 4.00 GB RAM size and processor speed of 2.40 GHz.

CHAPTER V: RESULTS

Once the application has been opened, the main window appears after a splash screen. As shown in the figure below, it consists of icons which indicate the current step of the process, a help button, a short description of the step, the working area, status bar, and navigation buttons.



MaSOM: Clustering Microarray Data Using Self-Organizing Maps

Submit

Import files containing the expression ratios. Supplementary data for genes and samples may optionally be supplied for the documentation of results. All files must follow the format specified in the guidelines.

Choose files to process.

Data file*

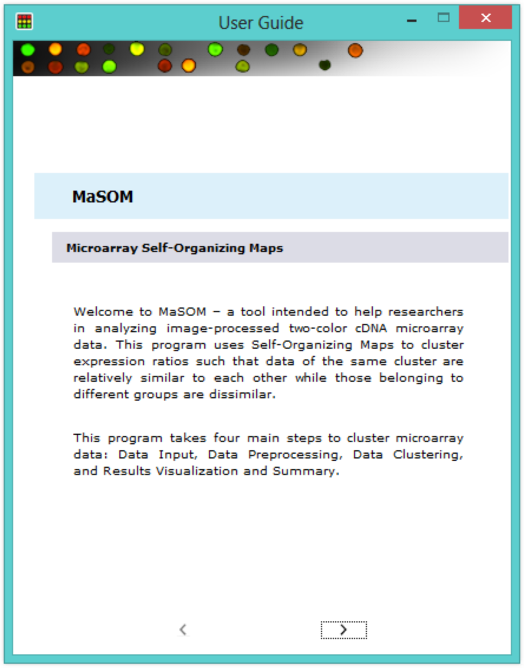
Gene description file

Sample description file

* Required information

Figure 11: Page 1 of main window, MaSOM

If unfamiliar with the tool, the user may open the User Guide, as shown in Figure 12, by clicking the help button or any help icons next to some fields in the working area. The User Guide window contains descriptions of the steps and the information it requires from the user.


User Guide

MaSOM

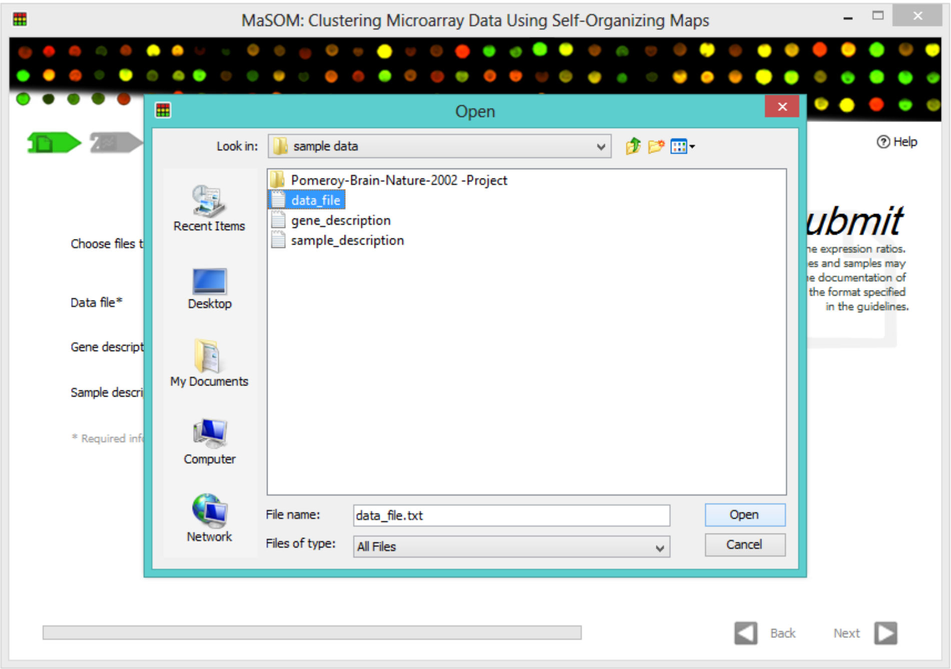
Microarray Self-Organizing Maps

Welcome to MaSOM – a tool intended to help researchers in analyzing image-processed two-color cDNA microarray data. This program uses Self-Organizing Maps to cluster expression ratios such that data of the same cluster are relatively similar to each other while those belonging to different groups are dissimilar.

This program takes four main steps to cluster microarray data: Data Input, Data Preprocessing, Data Clustering, and Results Visualization and Summary.

Figure 12: Help window, MaSOM

In order to proceed to the next step, the user must submit the data file and may optionally provide a description file for the samples or genes or both.


MaSOM: Clustering Microarray Data Using Self-Organizing Maps

Open

Look in: sample data

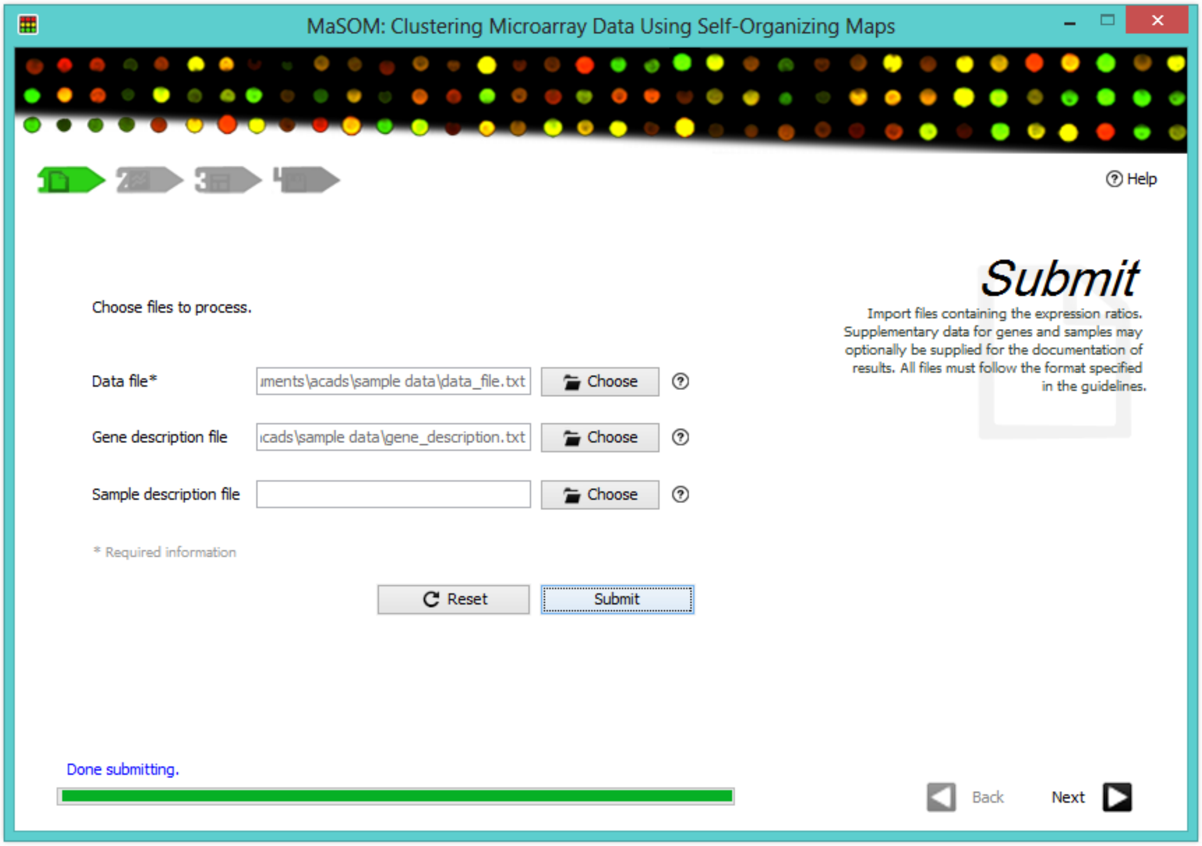
Pomeroy-Brain-Nature-2002 -Project
data_file
gene_description
sample_description

File name: data_file.txt

Files of type: All Files

Figure 13: Choosing data file, MaSOM

The chosen files are read by the tool upon clicking the Submit button. As it finishes reading, the user may proceed to the next step by clicking the Next button.


MaSOM: Clustering Microarray Data Using Self-Organizing Maps
Help
Choose
Choose
Choose
Reset
Submit
Back
Next

Submit

Import files containing the expression ratios. Supplementary data for genes and samples may optionally be supplied for the documentation of results. All files must follow the format specified in the guidelines.

Choose files to process.

Data file* ıments\acads\sample data\data_file.txt

Gene description file ıcads\sample data\gene_description.txt

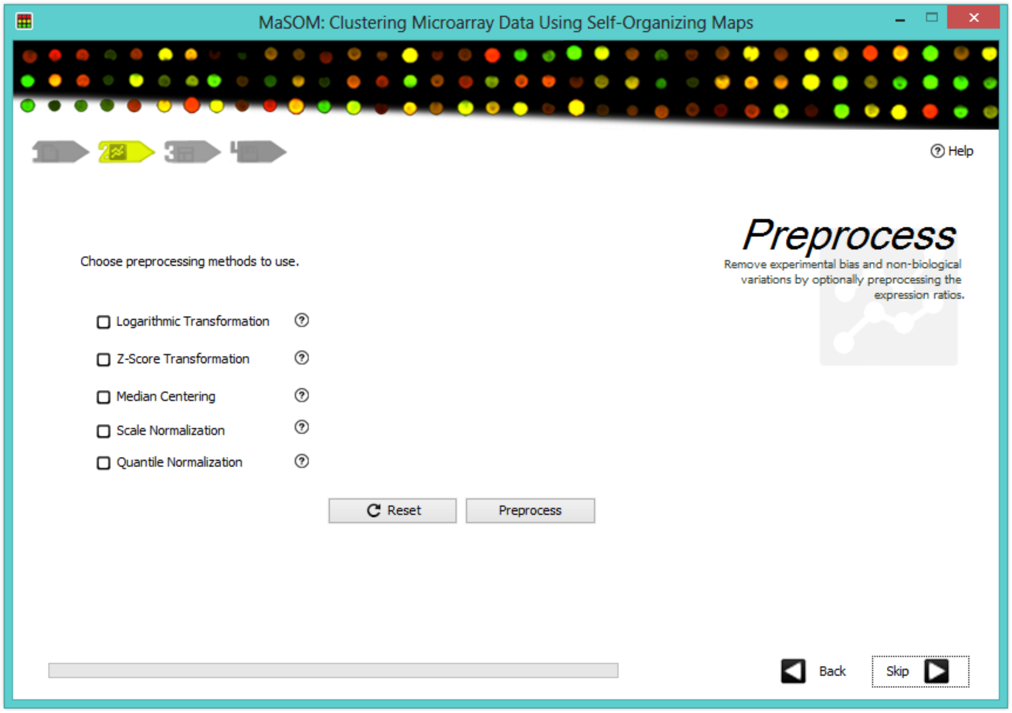
Sample description file

* Required information

Done submitting.

Figure 14: Submitting files, MaSOM

The next figure shows the second step which is for the preprocessing of the microarray data. In this phase, the user is given the option to preprocess the expression ratios with the available preprocessing techniques namely, logarithmic transformation, z-score transformation or constant normalization, median centering, scale normalization, and quantile normalization. Help icons are available for each method for a short description on what it does on the data. The user may choose one or more technique, as shown in Figure 16, or may simply skip the step if the data does not need to be preprocessed.


MaSOM: Clustering Microarray Data Using Self-Organizing Maps
Help
Reset
Preprocess
Back
Skip

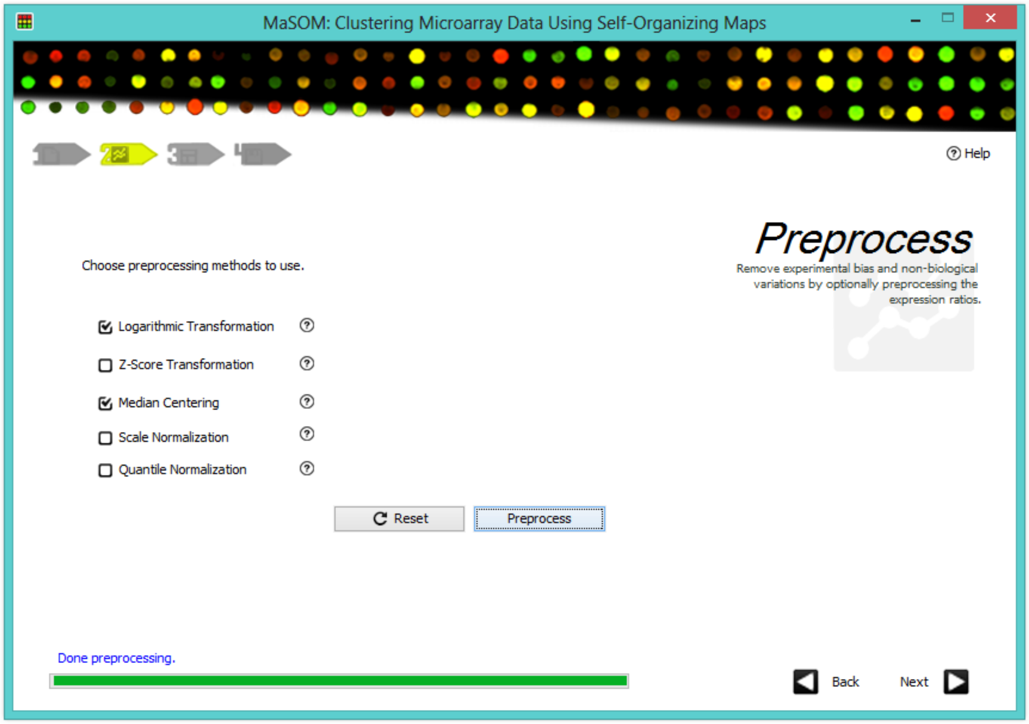
Preprocess

Remove experimental bias and non-biological variations by optionally preprocessing the expression ratios.

Choose preprocessing methods to use.

- ☐ Logarithmic Transformation
- ☐ Z-Score Transformation
- ☐ Median Centering
- ☐ Scale Normalization
- ☐ Quantile Normalization

Figure 15: Page 2 of main window, MaSOM


MaSOM: Clustering Microarray Data Using Self-Organizing Maps
Help
Reset
Preprocess
Back
Next

Preprocess

Remove experimental bias and non-biological variations by optionally preprocessing the expression ratios.

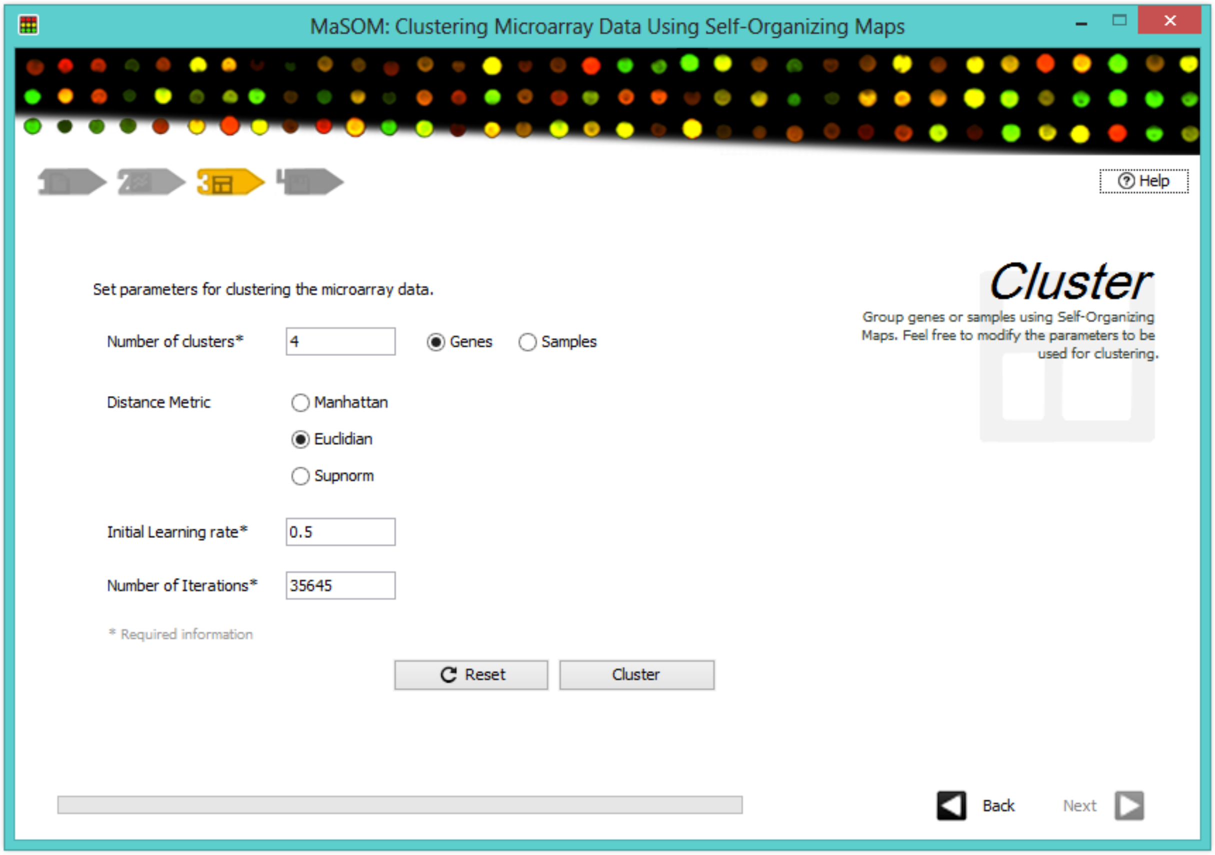
Choose preprocessing methods to use.

- ☑ Logarithmic Transformation
- ☐ Z-Score Transformation
- ☑ Median Centering
- ☐ Scale Normalization
- ☐ Quantile Normalization

Done preprocessing.

Figure 16: Preprocessing data, MaSOM

Figure 17 shows the clustering step. All values required by the form are given default values. By default, the number of iterations is five times the number of data points to cluster. Values may be changed depending on the preference of the user as shown in Figure 18. As soon as the clustering is done, the user may be able to proceed to the next step to view the results.



MaSOM: Clustering Microarray Data Using Self-Organizing Maps

Cluster

Group genes or samples using Self-Organizing Maps. Feel free to modify the parameters to be used for clustering.

Set parameters for clustering the microarray data.

Number of clusters* 4 ● Genes ○ Samples

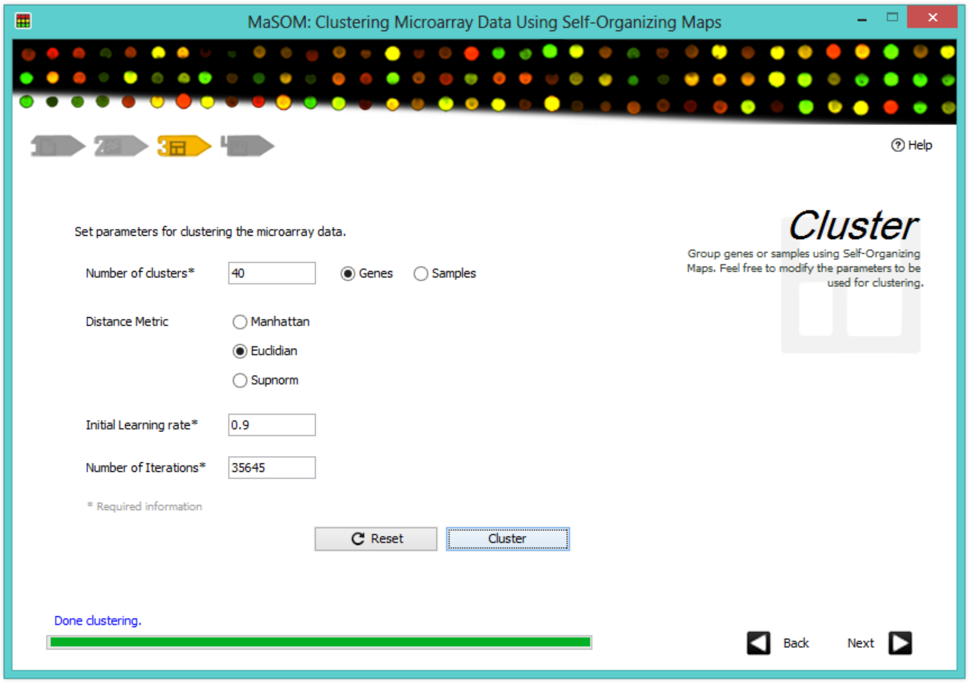
Distance Metric ○ Manhattan ● Euclidian ○ Supnorm

Initial Learning rate* 0.5

Number of Iterations* 35645

* Required information

Figure 17: Page 3 of main window, MaSOM



MaSOM: Clustering Microarray Data Using Self-Organizing Maps

Cluster

Group genes or samples using Self-Organizing Maps. Feel free to modify the parameters to be used for clustering.

Set parameters for clustering the microarray data.

Number of clusters* 40 ● Genes ○ Samples

Distance Metric ○ Manhattan ● Euclidian ○ Supnorm

Initial Learning rate* 0.9

Number of Iterations* 35645

* Required information

Done clustering.

Figure 18: Clustering data, MaSOM

Once the Next button has been pressed, the tool displays the Result window. An example is shown in the next figures.

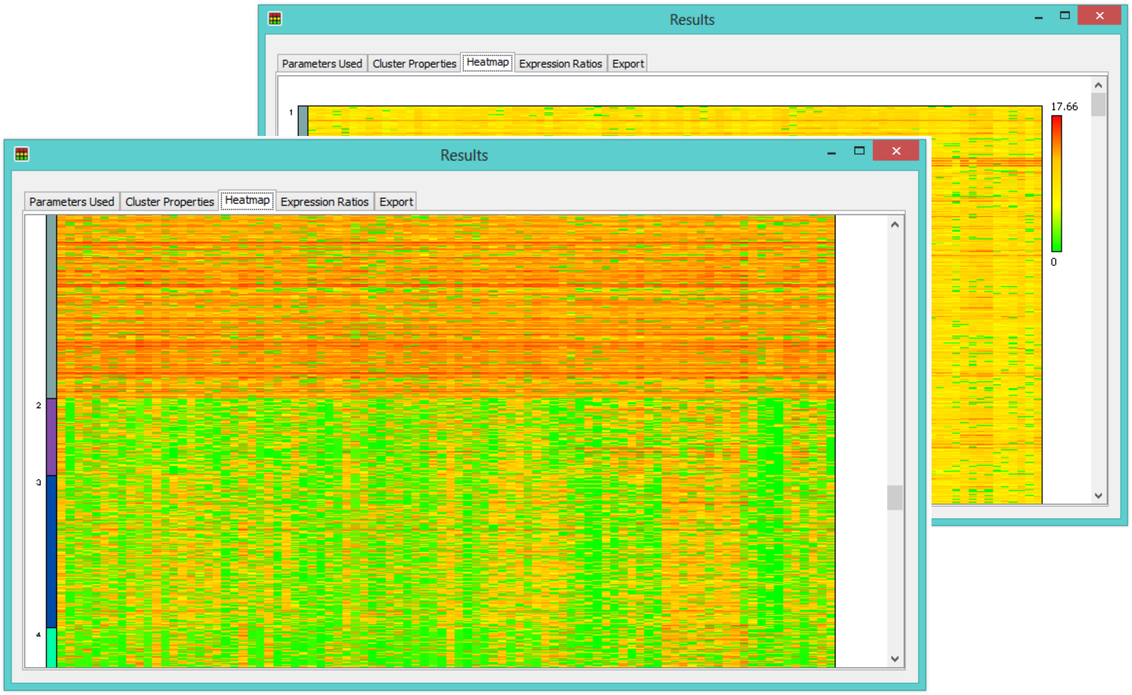
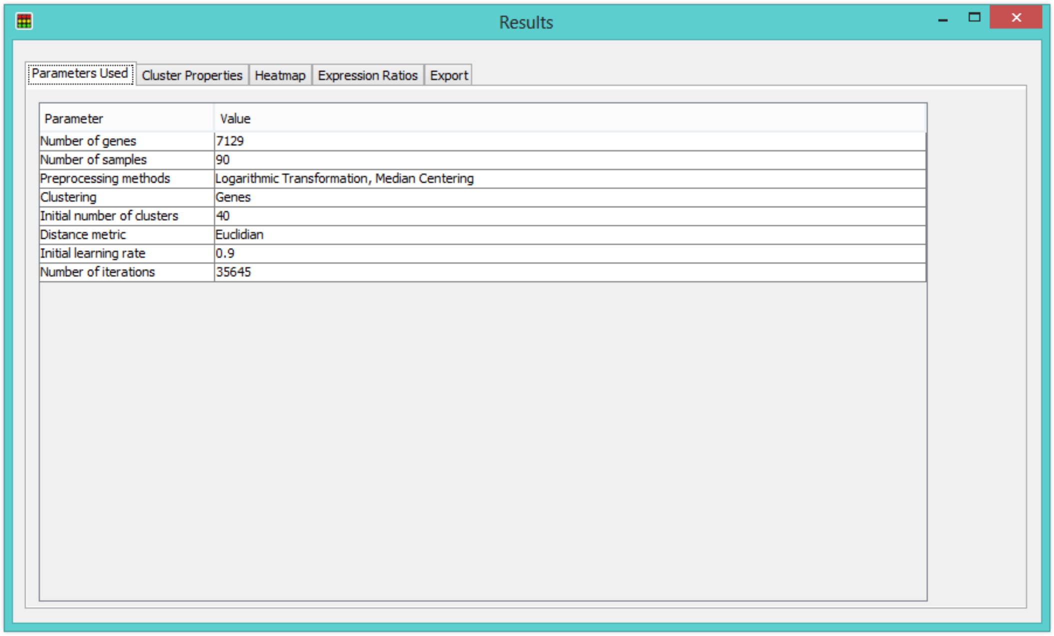


Figure 19: Heatmap, MaSOM

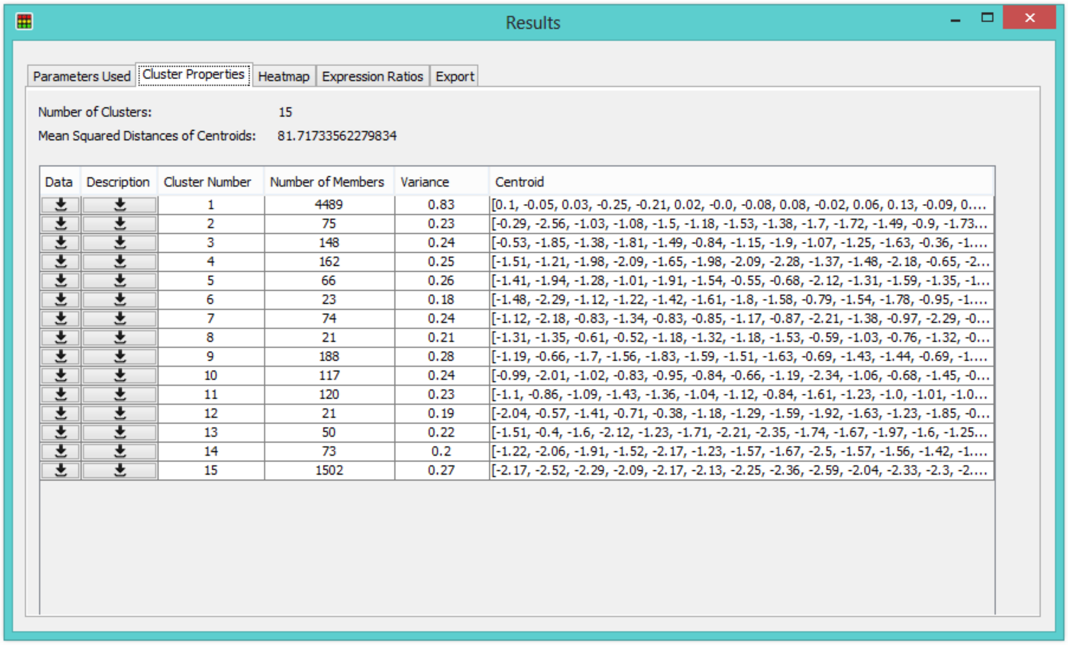
The figure above shows the resulting heatmap of the clustered data. It shows the legend which indicates the range of values and the color gradients which indicates the value of each data point with respect to the minimum and maximum values in the entire expression matrix. Data points that are labeled yellow are those located at the middle of the range of values or, interpreting the data, these are genes that did not change in expression level from the normal to the infected samples. Whereas the data points colored in red and green are being considered as up-regulated and down-regulated genes, respectively.

The next figures are the other components of Results window. It consists of three tables, namely, Parameters Used, Cluster Properties, and Expression Ratios. Each are shown the next three figures.



Parameter	Value
Number of genes	7129
Number of samples	90
Preprocessing methods	Logarithmic Transformation, Median Centering
Clustering	Genes
Initial number of clusters	40
Distance metric	Euclidian
Initial learning rate	0.9
Number of iterations	35645

Figure 20: Parameters Used, MaSOM

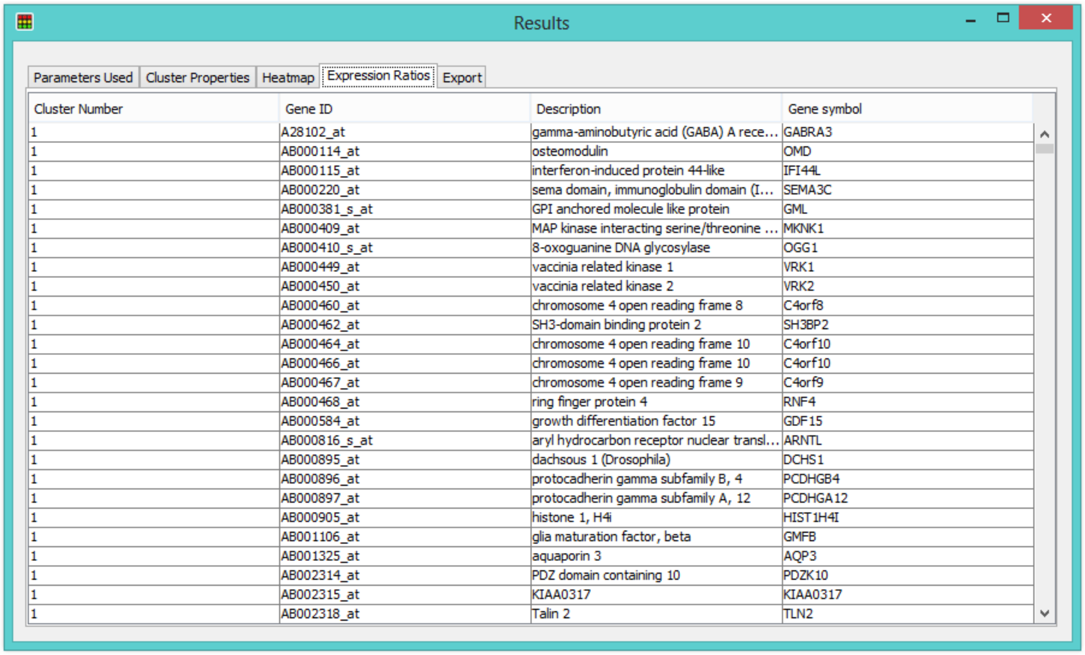


Number of Clusters: 15

Mean Squared Distances of Centroids: 81.71733562279834

Data	Description	Cluster Number	Number of Members	Variance	Centroid
⤓	⤓	1	4489	0.83	[0.1, -0.05, 0.03, -0.25, -0.21, 0.02, -0.0, -0.08, 0.08, -0.02, 0.06, 0.13, -0.09, 0....
⤓	⤓	2	75	0.23	[-0.29, -2.56, -1.03, -1.08, -1.5, -1.18, -1.53, -1.38, -1.7, -1.72, -1.49, -0.9, -1.73...
⤓	⤓	3	148	0.24	[-0.53, -1.85, -1.38, -1.81, -1.49, -0.84, -1.15, -1.9, -1.07, -1.25, -1.63, -0.36, -1....
⤓	⤓	4	162	0.25	[-1.51, -1.21, -1.98, -2.09, -1.65, -1.98, -2.09, -2.28, -1.37, -1.48, -2.18, -0.65, -2...
⤓	⤓	5	66	0.26	[-1.41, -1.94, -1.28, -1.01, -1.91, -1.54, -0.55, -0.68, -2.12, -1.31, -1.59, -1.35, -1...
⤓	⤓	6	23	0.18	[-1.48, -2.29, -1.12, -1.22, -1.42, -1.61, -1.8, -1.58, -0.79, -1.54, -1.78, -0.95, -1....
⤓	⤓	7	74	0.24	[-1.12, -2.18, -0.83, -1.34, -0.83, -0.85, -1.17, -0.87, -2.21, -1.38, -0.97, -2.29, -0...
⤓	⤓	8	21	0.21	[-1.31, -1.35, -0.61, -0.52, -1.18, -1.32, -1.18, -1.53, -0.59, -1.03, -0.76, -1.32, -0...
⤓	⤓	9	188	0.28	[-1.19, -0.66, -1.7, -1.56, -1.83, -1.59, -1.51, -1.63, -0.69, -1.43, -1.44, -0.69, -1....
⤓	⤓	10	117	0.24	[-0.99, -2.01, -1.02, -0.83, -0.95, -0.84, -0.66, -1.19, -2.34, -1.06, -0.68, -1.45, -0...
⤓	⤓	11	120	0.23	[-1.1, -0.86, -1.09, -1.43, -1.36, -1.04, -1.12, -0.84, -1.61, -1.23, -1.0, -1.01, -1.0...
⤓	⤓	12	21	0.19	[-2.04, -0.57, -1.41, -0.71, -0.38, -1.18, -1.29, -1.59, -1.92, -1.63, -1.23, -1.85, -0...
⤓	⤓	13	50	0.22	[-1.51, -0.4, -1.6, -2.12, -1.23, -1.71, -2.21, -2.35, -1.74, -1.67, -1.97, -1.6, -1.25...
⤓	⤓	14	73	0.2	[-1.22, -2.06, -1.91, -1.52, -2.17, -1.23, -1.57, -1.67, -2.5, -1.57, -1.56, -1.42, -1....
⤓	⤓	15	1502	0.27	[-2.17, -2.52, -2.29, -2.09, -2.17, -2.13, -2.25, -2.36, -2.59, -2.04, -2.33, -2.3, -2....

Figure 21: Cluster Properties, MaSOM

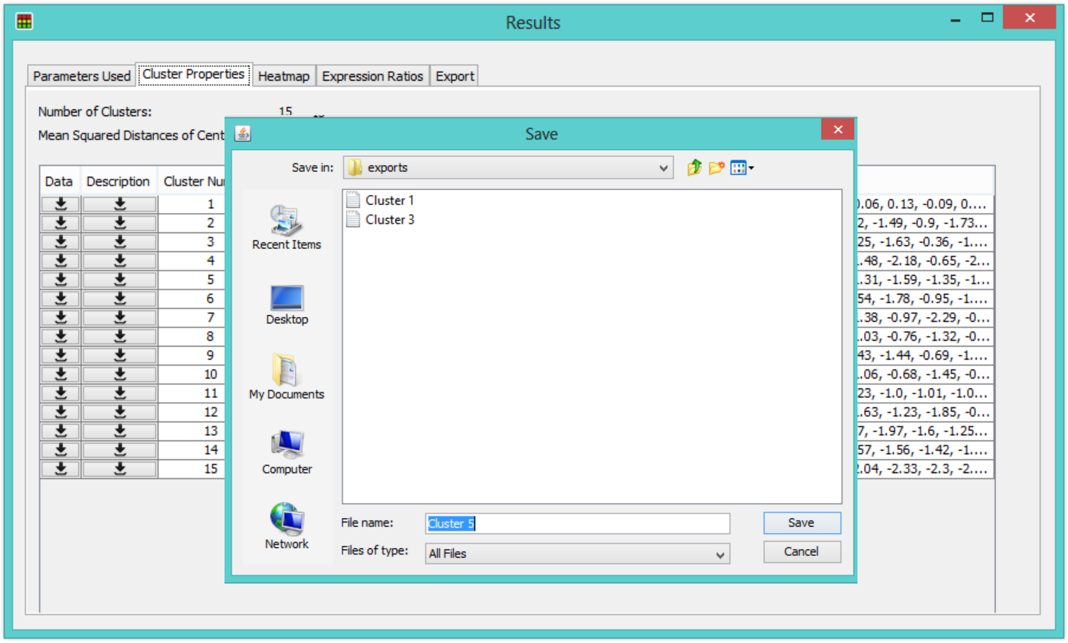


Cluster Number	Gene ID	Description	Gene symbol
1	A28102_at	gamma-aminobutyric acid (GABA) A rece...	GABRA3
1	AB000114_at	osteomodulin	OMD
1	AB000115_at	interferon-induced protein 44-like	IFI44L
1	AB000220_at	sema domain, immunoglobulin domain (I...	SEMA3C
1	AB000381_s_at	GPI anchored molecule like protein	GML
1	AB000409_at	MAP kinase interacting serine/threonine ...	MKNK1
1	AB000410_s_at	8-oxoguanine DNA glycosylase	OGG1
1	AB000449_at	vaccinia related kinase 1	VRK1
1	AB000450_at	vaccinia related kinase 2	VRK2
1	AB000460_at	chromosome 4 open reading frame 8	C4orf8
1	AB000462_at	SH3-domain binding protein 2	SH3BP2
1	AB000464_at	chromosome 4 open reading frame 10	C4orf10
1	AB000466_at	chromosome 4 open reading frame 10	C4orf10
1	AB000467_at	chromosome 4 open reading frame 9	C4orf9
1	AB000468_at	ring finger protein 4	RNF4
1	AB000584_at	growth differentiation factor 15	GDF15
1	AB000816_s_at	aryl hydrocarbon receptor nuclear transl...	ARNTL
1	AB000895_at	dachsous 1 (Drosophila)	DCHS1
1	AB000896_at	protocadherin gamma subfamily B, 4	PCDHGB4
1	AB000897_at	protocadherin gamma subfamily A, 12	PCDHGA12
1	AB000905_at	histone 1, H4i	HIST1H4I
1	AB001106_at	glia maturation factor, beta	GMFB
1	AB001325_at	aquaporin 3	AQP3
1	AB002314_at	PDZ domain containing 10	PDZK10
1	AB002315_at	KIAA0317	KIAA0317
1	AB002318_at	Talin 2	TLN2

Figure 22: Expression Ratios, MaSOM

The first table displays the values set by the user or the system that were used during the preceding steps of the process. This includes the dimension of the data, preprocessing methods, and the parameters for the clustering. The second one enumerates the clusters including the number of members, variance, and centroid of each of them. In this table, the user is given a means to export the gene expressions or the description, if available, of the members of a specific cluster as shown in Figure 23. This allows the user to perform subclustering by submitting the exported data files to the tool and performing another clustering to it. In this manner, the user is given an option to adjust the parameters of the clustering depending on the perceived appropriate values for each subcluster.

The last table is called Expression Ratios which displays the identification keys of the sample or genes together with their clusters. If the user has submitted a description file for the data during the first step, the descriptions for each of the clustered data are enumerated in order to guide and help the user in associating external information when analyzing the results.



Number of Clusters: 15

Mean Squared Distances of Cent

Data	Description	Cluster Nu	
↓	↓	1	).06, 0.13, -0.09, 0....
↓	↓	2	2, -1.49, -0.9, -1.73...
↓	↓	3	25, -1.63, -0.36, -1....
↓	↓	4	.48, -2.18, -0.65, -2...
↓	↓	5	.31, -1.59, -1.35, -1...
↓	↓	6	54, -1.78, -0.95, -1....
↓	↓	7	.38, -0.97, -2.29, -0...
↓	↓	8	.03, -0.76, -1.32, -0...
↓	↓	9	43, -1.44, -0.69, -1....
↓	↓	10	.06, -0.68, -1.45, -0...
↓	↓	11	23, -1.0, -1.01, -1.0...
↓	↓	12	.63, -1.23, -1.85, -0...
↓	↓	13	7, -1.97, -1.6, -1.25...
↓	↓	14	57, -1.56, -1.42, -1....
↓	↓	15	.04, -2.33, -2.3, -2....

Save in: exports

Cluster 1

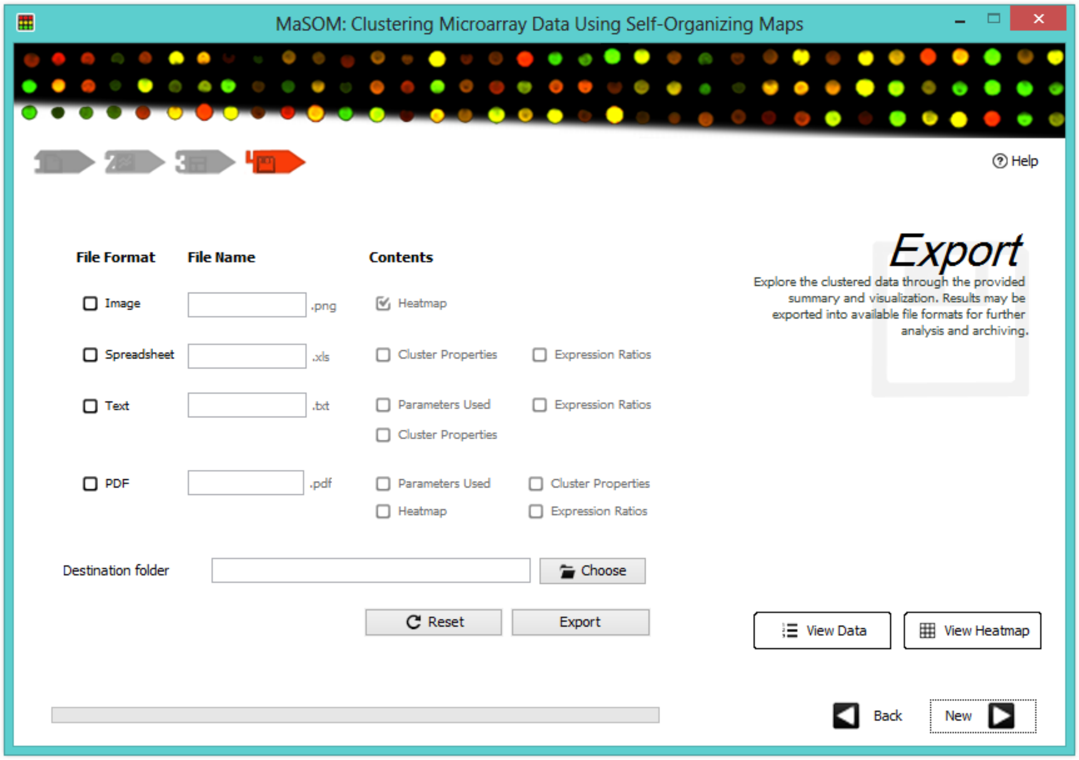
Cluster 3

File name: Cluster 5

Files of type: All Files

Figure 23: Exporting a specific cluster, MaSOM

The last step of the process, which is exporting of the results, is shown in the figure below. In this phase the user is given the option to save the most recent result in various formats depending on what contents are to be exported.



MaSOM: Clustering Microarray Data Using Self-Organizing Maps

Export

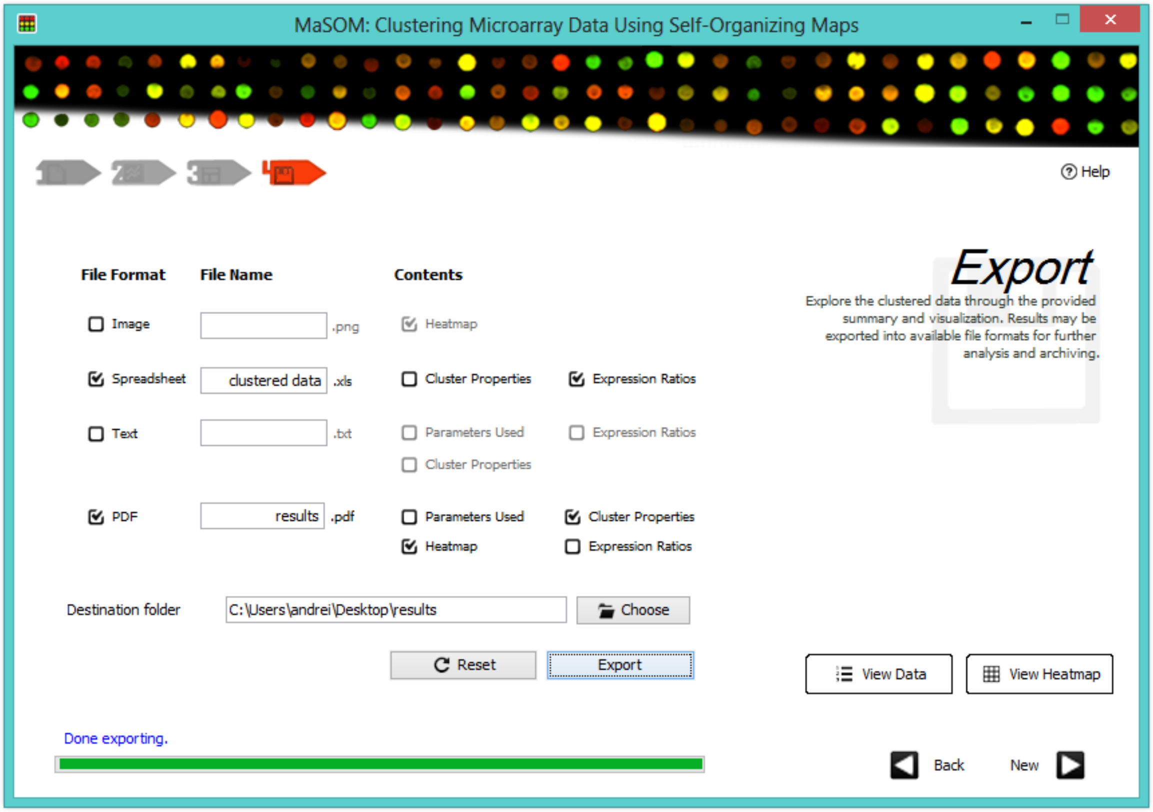
Explore the clustered data through the provided summary and visualization. Results may be exported into available file formats for further analysis and archiving.

File Format	File Name	Contents	
□ Image	.png	☑ Heatmap	
□ Spreadsheet	.xls	□ Cluster Properties	□ Expression Ratios
□ Text	.txt	□ Parameters Used □ Cluster Properties	□ Expression Ratios
□ PDF	.pdf	□ Parameters Used □ Heatmap	□ Cluster Properties □ Expression Ratios

Destination folder

Figure 24: Page 4 of main window, MaSOM

After the user has chosen from image, spreadsheet, text, and PDF what format and contents to be saved, the tool then saves the data into the chosen directory. This process is illustrated by Figures 25 and portions of the exported data are shown in Figures 26 to 28.



MaSOM: Clustering Microarray Data Using Self-Organizing Maps

Export

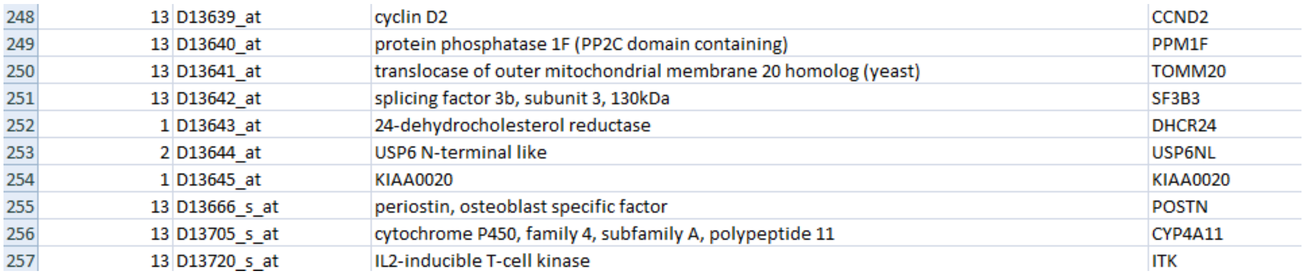
Explore the clustered data through the provided summary and visualization. Results may be exported into available file formats for further analysis and archiving.

File Format	File Name	Contents	
☐ Image	.png	☑ Heatmap	
☑ Spreadsheet	clustered data .xls	☐ Cluster Properties	☑ Expression Ratios
☐ Text	.txt	☐ Parameters Used ☐ Cluster Properties	☐ Expression Ratios
☑ PDF	results .pdf	☐ Parameters Used ☑ Heatmap	☑ Cluster Properties ☐ Expression Ratios

Destination folder C:\Users\andrei\Desktop\results

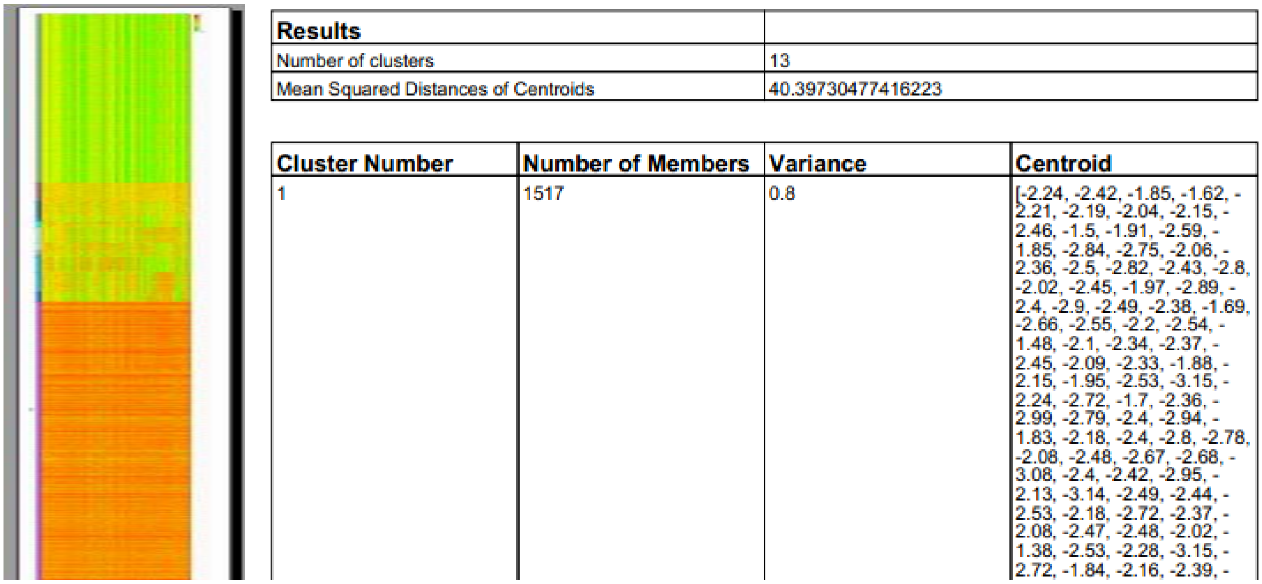
Done exporting.

Figure 25: Exporting results, MaSOM



248	13	D13639_at	cyclin D2	CCND2
249	13	D13640_at	protein phosphatase 1F (PP2C domain containing)	PPM1F
250	13	D13641_at	translocase of outer mitochondrial membrane 20 homolog (yeast)	TOMM20
251	13	D13642_at	splicing factor 3b, subunit 3, 130kDa	SF3B3
252	1	D13643_at	24-dehydrocholesterol reductase	DHCR24
253	2	D13644_at	USP6 N-terminal like	USP6NL
254	1	D13645_at	KIAA0020	KIAA0020
255	13	D13666_s_at	periostin, osteoblast specific factor	POSTN
256	13	D13705_s_at	cytochrome P450, family 4, subfamily A, polypeptide 11	CYP4A11
257	13	D13720_s_at	IL2-inducible T-cell kinase	ITK

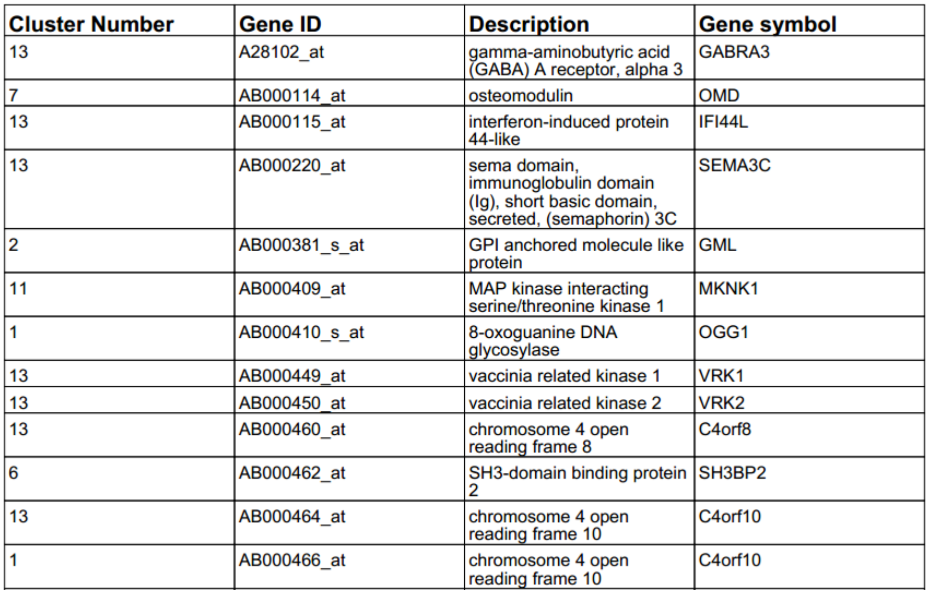
Figure 26: Exported spreadsheet of clustered data, MaSOM



Results	
Number of clusters	13
Mean Squared Distances of Centroids	40.39730477416223

Cluster Number	Number of Members	Variance	Centroid
1	1517	0.8	[-2.24, -2.42, -1.85, -1.62, -2.21, -2.19, -2.04, -2.15, -2.46, -1.5, -1.91, -2.59, -1.85, -2.84, -2.75, -2.06, -2.36, -2.5, -2.82, -2.43, -2.8, -2.02, -2.45, -1.97, -2.89, -2.4, -2.9, -2.49, -2.38, -1.69, -2.66, -2.55, -2.2, -2.54, -1.48, -2.1, -2.34, -2.37, -2.45, -2.09, -2.33, -1.88, -2.15, -1.95, -2.53, -3.15, -2.24, -2.72, -1.7, -2.36, -2.99, -2.79, -2.4, -2.94, -1.83, -2.18, -2.4, -2.8, -2.78, -2.08, -2.48, -2.67, -2.68, -3.08, -2.4, -2.42, -2.95, -2.13, -3.14, -2.49, -2.44, -2.53, -2.18, -2.72, -2.37, -2.08, -2.47, -2.48, -2.02, -1.38, -2.53, -2.28, -3.15, -2.72, -1.84, -2.16, -2.39, -

Figure 27: Exported heatmap and clusters, MaSOM



Cluster Number	Gene ID	Description	Gene symbol
13	A28102_at	gamma-aminobutyric acid (GABA) A receptor, alpha 3	GABRA3
7	AB000114_at	osteomodulin	OMD
13	AB000115_at	interferon-induced protein 44-like	IFI44L
13	AB000220_at	sema domain, immunoglobulin domain (Ig), short basic domain, secreted, (semaphorin) 3C	SEMA3C
2	AB000381_s_at	GPI anchored molecule like protein	GML
11	AB000409_at	MAP kinase interacting serine/threonine kinase 1	MKNK1
1	AB000410_s_at	8-oxoguanine DNA glycosylase	OGG1
13	AB000449_at	vaccinia related kinase 1	VRK1
13	AB000450_at	vaccinia related kinase 2	VRK2
13	AB000460_at	chromosome 4 open reading frame 8	C4orf8
6	AB000462_at	SH3-domain binding protein 2	SH3BP2
13	AB000464_at	chromosome 4 open reading frame 10	C4orf10
1	AB000466_at	chromosome 4 open reading frame 10	C4orf10

Figure 28: Exported data description, MaSOM

This process can also be done in the Results window through the use of the interface at the Export tab. This is convenient especially when the user has generated multiple results and a previous result is chosen to be exported.

CHAPTER VI: DISCUSSION

MaSOM or Microarray Self-Organizing Maps is a simple and user-friendly tool that primarily clusters expression ratios from microarray data with the use of Self-Organizing Maps. It is a four-step process that enables the user to preprocess, cluster, and export data. First of the processes is data submission wherein the user imports the data to cluster and next is the preprocessing of the data with which the user may choose to prepare the data using the available methods which are the Logarithmic Transformation, Z-Score Transformation or Constant Normalization, Median Centering, Scale Normalization, and Quantile Normalization.

The third is the main step which is clustering. In this phase the user is given the option to cluster microarray data by genes or by samples with the given initial number of clusters. The Self-Organizing Map will then group the data depending on the set parameters such as the initial learning rate, number of iterations, and the distance metric. The user may adjust these parameters as the results of this method changes at different variations of the parameters.

The last of the steps is where the results can be viewed, assessed, and exported. The tool displays the properties of the generated clusters and, if available, includes the properties or descriptions of each of the data points. With this, the user may decide which clustering parameters best lead to the most appropriate or desired result. The tool provides export facilities for archiving and referencing. Furthermore, specific clusters may also be exported into files, in a format that can be read by the tool, for further re-clustering. This somehow gives the user the ability to perform hierarchical clustering to the data in such a way that clustering parameters may still be adjusted to fit the grouping of a specific subcluster.

Clustering microarray data gives the practitioners an insight on how a large group of data is distributed. This helps them in choosing what statistical methods to use in their principal analysis of the data. The tool, which uses Self-Organizing Maps as a clustering method, thus contributes to the users' pre-assessment of the two-color cDNA microarray data being studied.

The tool's facility of visualizing its results is essential for microarray data as it aids in exploratory data analysis because it gives the users a larger view of the differences among the clusters. Furthermore, visualization helps researchers in analyzing large data especially in discovering relationships among the variables and examining the behavior of the data.

But this tool, as it uses and relies mostly on the Self-Organizing Maps, brings along the disadvantages of the method in its clustering. The results of SOM highly depend on the preprocessing methods used and parameters set by the user before the clustering process. Thus, it is on the users' discretion which result is the most suitable for their purpose of the clustering. Furthermore, when same parameters were set during different executions of the process, the tool outputs equivalent clusters but may possibly have different ordering, i.e., a resulting sequence of clusters may be inverted at another execution. This is due to the randomization of the initial vectors of the map. Lastly, the results of this tool may differ from the results of other existing clustering tools because of their differences in approach to the data. But these are defeated by the advantages of SOM compared to most clustering methods such as its ability to cluster data while preserving their natural topography or distribution and its capability to reduce the dimension of high-dimensional data, like those of the microarray, so that they become suitable to be visualized.

CHAPTER VII: CONCLUSION

MaSOM, a tool that clusters gene expression ratios from two-color cDNA microarray, relies on Self-Organizing Maps in order to group large and multi-dimensional data. It provides preprocessing methods to remove possible non-biological variances and experimental biases among the samples of the data. After clustering the data according to the parameters set for Self-Organizing Maps, the partitions are being visualized using a heatmap. This is also accompanied by tabular presentation of the results which may be exported into several file formats for the future use of these results.

This software can be used to cluster any dimension of available gene expression ratios and samples since its main objective is to discover the natural structure of the given large data set regardless of its data collection. It helps the practitioners in determining the initial properties of the data they study before proceeding to their actual experimentation onto their data. But this tool, as microarray data usually come in very large dimensions, requires considerably large memory and high processing speed for good performance. Since this tool basically clusters microarray data, developments are necessary whenever specific features or extensive analysis, such as those mentioned in the Recommendations, are sought.

CHAPTER VIII: RECOMMENDATIONS

Although MaSOM seems suitable for clustering gene expression ratios from cDNA microarray data, there are still features that can be further improved. One is the list of available preprocessing techniques. Though users may wish to preprocess the data using other software before clustering using the tool, it is a lot better if more techniques are available in the program itself. There are lots of methods, and variations of those methods, being used as microarray data preprocessing but not all are fitted to the tool's specified data contents and structure.

Self-Organizing Maps may also be reconfigured into its other variations. SOM's dimension may be one-, two-, or multi-dimensional and its shape may be rectangular or hexagonal. This tool used one-dimensional rectangular Self-Organizing Map so some experts may suggest developing it by utilizing other configurations of SOM as its clustering method. Furthermore, SOM may be re-constructed to allow hierarchical clustering so that the users would have a straightforward approach whenever subclustering is necessary for their study.

Lastly, there are several other means to visualize microarray data and it may be more beneficial to the users if they are given the option on which visualization technique is to be used. These techniques vary in what properties of the large data they emphasize. In particular, heatmaps focus on the clusters and the relative values of the data points which already perceived essential and suitable for the current purpose of this tool. Other methods may be needed especially when the tool extends its objectives.

CHAPTER IX: BIBLIOGRAPHY

[1] Sharma, L. K., & Rungta, S. (2012). Comparative Study of Data Cluster Analysis for Microarray. *International Journal of Computer Trends and Technology Vol. 3 Issue 3* , 387 - 390.

[2] Zhang, Y., Ding, C., & Li, T. (2008). Gene selection algorithm by combining reliefF and mRMR. *BMC Genomics*.

[3] Hewett, R., & Kijsanayothin, P. (2008). Tumor classification ranking from microarray data. *BMC Genomics*.

[4] Bhuvaneswari, V., & Brintha, S. (2012). Microarray Gene Expression Analysis Using Type 2 Fuzzy Logic (MGA-FL). *International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.2* , 53-69.

[5] Venkatesh, E., & Thangaraj, D. (2010). Investigation of Micro Array Gene Expression Using Linear Vector Quantization for Cancer. *(IJCSE) International Journal on Computer Science and Engineering* , 2114-2116.

[6] Emilyn, J. J., & Ramar, K. (2011). A Rough Set based Gene Expression Clustering Algorithm. *Journal of Computer Science 7* , 986-990.

[7] Richards, A. L., Holmans, P., O'Donovan, M. C., Owen, M. J., & Jones, L. (2008). A comparison of four clustering methods for brain expression microarray data. *BMC Bioinformatics*.

[8] Rani, L., & Rajalakshmi, P. (2012). Clustering Gene Expression Data using Quad Tree based Expectation Maximization Approach. *International Journal of Applied Information Systems (IJAIS) Vol. 2 No. 2* , 10-13.

[9] Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., et al. (1999). Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA* , 2907–2912.

[10] Vanichayobon, S., Wichaidit, S., & Wettayaprasit, W. (2007). Microarray Gene Selection Using Self-Organizing Map. *7th WSEAS International Conference on Simulation, Modelling and Optimization.* Beijing, China.

[11] Yi, C. H., & Hasan, Y. A. (2011). Microarray Data Mining with Fuzzy Self-Organising Maps. *Journal of Applied Mathematics & Bioinformatics, vol.1, no.1* , 159-174.

[12] Torkkola, K., Gardner, R. M., Kraysser-Kranich, T., & Ma, C. (2001). Self-organising maps in mining gene expression data. *Information Sciences* , 79-96.

[13] McGarry, K., Sarfraz, M., & MacIntyre, J. (2007). Integrating Gene Expression Data from Microarrays using the Self-Organising Map and the Gene Ontology. *PRIB'07 Proceedings of the*

2nd IAPR international conference on Pattern recognition in bioinformatics, (pp. 206-217). Springer-Verlag Berlin, Heidelberg.

[14] Scharl, T., Voglhuber, I., & Leisch, F. (2009). Exploratory and inferential analysis of gene cluster neighborhood graphs. *BMC Bioinformatics*.

[15] Hardin, J., Hoopes, L., & Murphy, R. (2007). Analyzing DNA Microarrays with Undergraduate Statisticians. *ICOTS-7*.

[16] Salem, D. A., Abul, R. A., & Ali, H. A. (2011). MGS-CM: A Multiple Scoring Gene Selection Technique for Cancer Classification using Microarrays. *International Journal of Computer Applications*.

[17] Babu, M. M. An Introduction to Microarray Data Analysis.

[18] Shannon, W., Culverhouse, R., & Duncan, J. (2003). Analyzing microarray data using cluster analysis. *Pharmacogenomics* , 41-52.

[19] *Kohonen's Self Organizing Feature Maps*. (n.d.). Retrieved October 13, 2012, from ai-junkie: http://www.ai-junkie.com/ann/som.

[20] Yang, Y. H., Dudoit, S., Luu, P., Lin, D., Peng, V., Ngai, J., et al. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research Vol. 3 No. 4.*

[21] Bolstad, B. M., Irizzary, R. A., Astrad, M., & Speed, T. P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *BioInformatics* , 185-193.

[22] Tarca, A. L., Romero, R., & Draghici, S. (2006). Analysis of microarray experiments of gene expression profiling. *American Journal of Obstetrics and Gynecology* , 373-388.

[23] Zhao, Y., & Simon, R. (2008). *BRB-ArrayTools Data Archive for Human Cancer Gene Expression: A Unique and Efficient Data Sharing Resource.* USA: Cancer Informatics.

CHAPTER X: APPENDIX

A. MaSOM/src/SOM.java

```
public class SOM {
       private SOM_node nodes[];
       public double initLearningRate;
       private double initMapRadius;
       private double timeConstant;
       public int clusters;
       public int maxIter;
       public int distMetric;
       public boolean isByCols;
       public double data[][];

      public SOM(double data[][], int clusters, int maxIter, int distMetric, double
        initLearningRate, boolean isByCols){
              this.data = new double[data.length][data[0].length];
              for (int iter = 0; iter < data.length; iter++)
                  this.data[iter] = data[iter].clone();
              this.clusters = clusters;
              this.maxIter = maxIter;
              this.distMetric = distMetric;
              this.initLearningRate = initLearningRate;
              this.initMapRadius = clusters/2;
              this.isByCols = isByCols;
              timeConstant = maxIter/ Math.log(initMapRadius);
              if(isByCols) this.data = invertArray(data,data[0].length);
              nodes = new SOM_node[clusters];
              initNodes(this.data[0].length);
       }
       private double getDistance(double x[], double y[]){
              double distance = 0;
              switch (distMetric) {
              case 1: // manhattan
                     for(int i=0; i< x.length; i++)
                            distance += Math.abs(x[i] - y[i]);
                     break;
              case 2: // euclidian
                     for(int i=0; i< x.length; i++)
                            distance += (x[i] - y[i]) * (x[i] - y[i]);
                     distance = Math.sqrt(distance);
                     break;
              case 3: // sup-norm
                     for(int i=0; i< x.length; i++)
                            distance = Math.max(Math.abs(x[i] - y[i]), distance);
                     break;
              default:
                     break;
              }
              return distance;
       }
       public double[][] invertArray(double originalData[][],int cols){
              double invData[][] = new double[cols][originalData.length];
              for(int i=0; i<originalData.length; i++)
                     for(int j=0; j<cols; j++)
                            invData[j][i]=originalData[i][j];
              return invData;
       }
       private void initNodes(int cols){
              for(int i=0; i<nodes.length; i++){
                     nodes[i] = new SOM_node(i,cols);
              }
       }
       private int getBMU(int dataIndex){
              int indexOfBMU=0;
              double minDistance=Double.POSITIVE_INFINITY;
              for(int i = 0; i<nodes.length; i++){
                     double distance =getDistance(data[dataIndex], nodes[i].weight);
                     if( distance < minDistance){
                            indexOfBMU = i;
                            minDistance = distance;
                     }
              }
              return indexOfBMU;
       }
       private double getLearningRate(int iter){
              return (initLearningRate*Math.exp(-iter/maxIter));
       }
       private double getNeighborhood(int nodeIndex,int BMUIndex,int iter){
              double radius = (initMapRadius*Math.exp(-iter/timeConstant));
              double nodeArr[]={nodeIndex}, bmuArr[]= {BMUIndex};
              double distance = getDistance(nodeArr, bmuArr );
              return ( Math.exp(-(distance*distance)/(2*radius*radius*iter)) );
       }
       public void train(){
              int iter = 1;
              double learningRate = 1;
              int node_WeightVectorSize = nodes[0].weight.length;
              while(iter < maxIter && learningRate > 0){
                     for(int dataIndex=0; dataIndex < data.length; dataIndex++,
                     iter++){// each data
                            int BMUIndex = getBMU(dataIndex);//best matching unit
                            learningRate = getLearningRate(iter);//learning rate
                            for(int nodeIndex=0; nodeIndex< nodes.length;
                               nodeIndex++){// each node of map
                                   double neighborhood =
                                         getNeighborhood(nodeIndex,BMUIndex,iter); //
                                         neighborhood
                                  for(int i=0; i< node_WeightVectorSize; i++)
                                         // each variable of the vector
                                         nodes[nodeIndex].weight[i] =
                                            nodes[nodeIndex].weight[i] +
                                            neighborhood*learningRate*(data[dataIndex][i
                                            ] - nodes[nodeIndex].weight[i]);
                            }
                     }
              }
       }
       public double[][] getCentroids(){
              int clusterSizes[] = new int[nodes.length];
              int numOfCluster=0;
              for(int dataIndex=0; dataIndex < data.length; dataIndex++)
                     clusterSizes[getBMU(dataIndex)]++;
              for(int i=0; i<clusterSizes.length;i++)
                     if(clusterSizes[i]>0) numOfCluster++;
              double centroids[][]=new double[numOfCluster][nodes[0].weight. length];
              for(int i=0,j=0; i<clusterSizes.length;i++)
                     if(clusterSizes[i]>0){
                            centroids[j]=nodes[i].weight;
                            j++;
                     }
              nodes = null;
              System.gc();
              return centroids;
       }
}
```

B. MaSOM/src/SOM_node.java

```
public class SOM_node {
       public double xCoord;
       public double weight[];

       public SOM_node(double x, int weightSize){
              xCoord = x;
              initializeWeight(weightSize);
       }
       private void initializeWeight(int weightSize){
              weight = new double[weightSize];
              for(int i=0; i<weight.length; i++)
                     weight[i]= Math.random();
       }
       public void setXCoord(double x){
              xCoord=x;
       }
       public double getXCoord(){
              return xCoord;
       }
}
```

C. MaSOM/src/KMeans.java

```
public class KMeans {
       public double clusters[][];
       public double data[][];
       public int numberofResultingClusters;
       public int distMetric;
       int clusterAssignment[];

       public KMeans(double initCentroids[][],double data[][], int distMetric){
              this.numberofResultingClusters = initCentroids.length;
              this.clusters=initCentroids;
              this.data = data;
              this.distMetric = distMetric;
              clusterAssignment = new int[data.length];
       }
       private double getDistance(double x[], double y[]){
              double distance = 0;
              switch (distMetric) {
              case 1: // euclidian
                     for(int i=0; i< x.length; i++)
                            distance += (x[i] - y[i]) * (x[i] - y[i]);
                     distance = Math.sqrt(distance);
                     break;
              case 2: // manhattan
                     for(int i=0; i< x.length; i++)
                            distance += Math.abs(x[i] - y[i]);
                     break;
              case 3: // sup-norm
                     for(int i=0; i< x.length; i++)
                            distance = Math.max(Math.abs(x[i] - y[i]), distance);
                     break;
```

```
        default:
            break;
        }
        return distance;
    }

    private int getCluster(int dataIndex){
        int indexOfClust=0;
        double minDistance=Double.POSITIVE_INFINITY;
        for(int i = 0; i<clusters.length; i++){
            double distance =getDistance(data[dataIndex], clusters[i]);
            if(distance < minDistance){
                indexOfClust = i;
                minDistance = distance;
            }
        }
        return indexOfClust;
    }

    public int[] groupData(){
        boolean didClusterChange=true;
        int clusters_weightSize =clusters[0].length;

        while(didClusterChange){
            didClusterChange = false;
            for(int dataIndex=0; dataIndex < data.length; dataIndex++)
                clusterAssignment[dataIndex] = getCluster(dataIndex);
            for(int clustIndex=0; clustIndex < clusters.length; clustIndex++){
                int size=0;
                double sum[]= new double[clusters_weightSize];
                for(int dataIndex=0; dataIndex < data.length; dataIndex++)
                    if(clusterAssignment[dataIndex]==clustIndex){
                        size++;
                        for(int i=0; i<clusters_weightSize; i++)
                            sum[i] += data[dataIndex][i];
                    }
                if (size>0){
                    for(int i=0; i<clusters_weightSize; i++){
                        double temp = clusters[clustIndex][i];
                        clusters[clustIndex][i] = sum[i]/size;
                        if(clusters[clustIndex][i] != temp)
                            didClusterChange = true;
                    }
                }else{
                    for(int i=0; i<clusters_weightSize; i++)
                        clusters[clustIndex][i]=Double.NaN;
                }
            }
        }
        for(int clustIndex = 0; clustIndex<clusters.length; clustIndex++)
            if(Double.isNaN(clusters[clustIndex][0]))
                numberofResultingClusters--;
        return clusterAssignment;
    }

    public double[][] arrangeDataIntoGroups(){
        int rows = data.length;
        int cols = data[0].length;
        double tempData[][] = new double[rows][cols];
        int index = 0;
        for(int clust = 0; clust<clusters.length; clust++)
            for(int i = 0; i<rows; i++)
                if(clusterAssignment[i]==clust){
```

```
                                        for(int j = 0; j<cols; j++)
                                                tempData[index][j]= data[i][j];
                                        index++;
                                }
                return tempData;
        }
}
```

D. MaSOM/src/HeatMap.java

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Arrays;

import javax.swing.*;

/* Modified Matthew Beckler's (matthew@mbeckler.org)
 * publicly and freely available code, HeatMap.java.
 */
@SuppressWarnings("serial")
public class HeatMap extends JPanel{
        private double[][] data;
        private int[][] dataColorIndices;
        private double largest ,smallest;
        private int rangeW = 20;
        private boolean clusterSamples;
        private int[] clusteringAssignments;
        private ArrayList<Integer> labelsX, labelsY;
        private Color[] labelColors;
        public BufferedImage bufferedImageLabel;
        public Graphics2D bufferedGraphicsLabel;
        public String title;
        private String xAxis = "Samples";
        private String yAxis = "Genes";
        private Color[] colors;
        private Color bg = Color.white;
        private Color fg = Color.black;
        public BufferedImage bufferedImage;
        public Graphics2D bufferedGraphics;

        public HeatMap(double[][] data, int[] clusteringAssignments, boolean
        clusterSamples, boolean useGraphicsYAxis, Color[] colors){
                super();
                labelsX = new ArrayList<>();
                labelsY = new ArrayList<>();
                labelsX.add(0);
                labelsY.add(0);
                this.colors = (Color[]) colors.clone();
                this.clusteringAssignments = clusteringAssignments;
                Arrays.sort(this.clusteringAssignments);
                this.clusterSamples = clusterSamples;
                this.data = new double[data.length][data[0].length];
                for (int ix = 0; ix < data.length; ix++){
                        for (int iy = 0; iy < data[0].length; iy++){
                                if (useGraphicsYAxis)this.data[ix][iy]= data[ix][iy];
                                else this.data[ix][iy]=data[ix][data[0].length-iy-1];
                        }
                }
```

```
       updateDataColors();
       this.setPreferredSize(new Dimension(60+data[0].length + rangeW,
       60+data.length));
       this.setDoubleBuffered(true);
       drawData();
}

/**
 * This uses the current array of colors that make up the gradient,
 * and assigns a color index to each data point, stored in
 * the dataColorIndices array, which is used by the
 * drawData() method to plot the points.
 */
private void updateDataColors(){
       largest = Double.MIN_VALUE;
       smallest = Double.MAX_VALUE;
       for (int x = 0; x < data.length; x++){
              for (int y = 0; y < data[0].length; y++){
                     largest = Math.max(data[x][y], largest);
                     smallest = Math.min(data[x][y], smallest);
              }
       }
       double range = largest - smallest;
       // dataColorIndices is the same size as the data array
       // It stores an int index into the color array
       dataColorIndices = new int[data.length][data[0].length];
       //assign a Color to each data point
       for (int x = 0; x < data.length; x++){
              for (int y = 0; y < data[0].length; y++){
                     double norm=(data[x][y]-smallest)/range;//0<norm<1
                     int colorIndex=(int)Math.floor(norm*
                     (colors.length-1));
                     dataColorIndices[x][y] = colorIndex;
              }
       }
       // assign color for cluster labels
       if(clusterSamples) labelColors = new Color[data[0].length];
       else labelColors = new Color[data.length];
       int prev = 0;
       float r=(float) .5, g=(float) 0.65, b=(float) 0.65;
       float d = (float) clusteringAssignments.length;// .15;
       while (d>=1) d/=10;
       for (int i = 0; i < clusteringAssignments.length; i++){
              if(clusteringAssignments[i]!= prev){
                     prev = clusteringAssignments[i];
                     if(prev%3 == 1){
                            g-=d/2;
                            if(g<0) g =1;
                     }else if(prev%3 == 2){
                            r+=d;
                            if(r>1)      r=0;
                     }else{
                            b-=d*2;
                            if(b<0) b =1;
                     }
                     if(clusterSamples)labelsX.add(i);
                     else labelsY.add(i);
              }
              labelColors[i] = new Color(r,g,b);
       }
}
/**
 * Creates a BufferedImage of the actual data plot.
 * After doing some profiling, it was discovered that 90% of the
 * drawing time was spent drawing the actual data (not on the axes or
 * tick marks).Since the Graphics2D has a drawImage method that can do
 * scaling, we are using that instead of scaling it ourselves. We only
 * need to draw the data into the bufferedImage on startup, or if the
 * data or gradient changes. This saves us an enormous amount of time.
 * Thanks to Josh Hayes-Sheen (grey@grevian.org) for the suggestion and
 * initial code to use the BufferedImage technique.
 * Since the scaling of the data plot will be handled by the drawImage
 * in paintComponent, we take the easy way out and draw our
 * bufferedImage with 1 pixel per data point. Too bad there isn't a
 * setPixel method in the Graphics2D class, it seems a bit silly to
 * fill a rectangle just to set a single pixel... This function
 * should be called whenever the data or the gradient changes.
 */
private void drawData(){
      bufferedImage = new BufferedImage(data[0].length,data.length,
      BufferedImage.TYPE_INT_ARGB);
      bufferedGraphics = bufferedImage.createGraphics();
      for (int x = 0; x < data.length; x++){
            for (int y = 0; y < data[0].length; y++){
            bufferedGraphics.setColor(colors[dataColorIndices[x][y]]);
                  bufferedGraphics.fillRect(y, x, 1, 1);
            }
      }
      // for the label
      if(clusterSamples)bufferedImageLabel = new BufferedImage(
data[0].length, 10 , BufferedImage.TYPE_INT_ARGB);
      else bufferedImageLabel = new BufferedImage( 10 ,data.length,
      BufferedImage.TYPE_INT_ARGB);
      bufferedGraphicsLabel = bufferedImageLabel.createGraphics();
      if(clusterSamples){
            for (int x = 0; x < clusteringAssignments.length; x++){
                  bufferedGraphicsLabel.setColor(labelColors[x]);
                  bufferedGraphicsLabel.fillRect(x, 0,1, 10);
            }
      }else{
            for (int y = 0; y < clusteringAssignments.length; y++){
                  bufferedGraphicsLabel.setColor(labelColors[y]);
                  bufferedGraphicsLabel.fillRect(0, y,10, 1);
            }
      }
}
/**
 * The overridden painting method, now optimized to simply draw the
 * data plot to the screen, letting the drawImage method do the
 * resizing. This saves an extreme amount of time.
 */
public void paintComponent(Graphics g){
      super.paintComponent(g);
      Graphics2D g2d = (Graphics2D) g;
      int width = this.getWidth();
      int height = this.getHeight();
      this.setOpaque(true);
      g2d.setColor(bg);// clear the panel
      g2d.fillRect(0, 0, width, height);
      if (bufferedImage == null)drawData();// draw the heat map
      // The data plot itself is drawn with 1 pixel per data point; the
      // drawImage method scales that up to fit our current window size
      g2d.drawImage(bufferedImage,
                  31, 31,
                  width - 30 - rangeW,
                  height - 30,
                0, 0,
                bufferedImage.getWidth(), bufferedImage.getHeight(),
                null);
        g2d.setColor(fg);// border
        g2d.drawRect(30, 30, width - 60 -rangeW, height - 60);
        // Y-Axis title
        //to get the text to fit nicely, we need to rotate the graphics
        g2d.rotate(Math.PI / 2);
        g2d.drawString(yAxis, (height / 2) - 4*yAxis.length(), -3);
        g2d.rotate( -Math.PI / 2); // X-Axis title
        g2d.drawString(xAxis,(width/2) - 4*xAxis.length(),height-3);
        int LabelH = 200;// Legend
        int maxY = 35, maxX = width - rangeW -20, rangeH =10;

        DecimalFormat numformat = new DecimalFormat("#.##");
        g2d.drawString(numformat.format(largest) , maxX, maxY);
        g2d.drawString(numformat.format(smallest), maxX, maxY+LabelH-40);
        g2d.drawRect(width - 20 - rangeW , 30+rangeH, 10, LabelH - 60);
        for (int y = 0; y < LabelH - 61; y++){
            int yStart ;
            yStart = LabelH - 31 +rangeH - y;
            g2d.setColor(colors[(int) ((y / (double) (LabelH - 60)) *
            (colors.length * 1.0))]);
            g2d.fillRect(width - 19 -rangeW, yStart, 9, 1);
        }
        // CLUSTER LABEL
        g2d.setFont(new Font("Arial", Font.PLAIN, 8));
        if(clusterSamples){
            int labelX = 30;
            int labelY = 20;
            g2d.setColor(fg);
            g2d.drawRect(labelX, labelY, width-60-rangeW, 10);
            g2d.drawImage(bufferedImageLabel, 31, 21, width - 30 -
            rangeW,30,0,0,bufferedImageLabel.getWidth(),
            bufferedImageLabel.getHeight(), null);
            g2d.setColor(new Color((float).1,(float).1,(float).1));
            for(int c=0; c< labelsX.size(); c++){
                int x =labelX+(labelsX.get(c)*(width-60-
                rangeW)/bufferedImageLabel.getWidth());
                g2d.drawString((c+1)+"" , x + 5, labelY-2);
                g2d.drawLine(x, labelY, x, labelY+10);
            }
        }else{
            int labelX = 20;
            int labelY = 30;
            g2d.setColor(fg);
            g2d.drawRect(labelX, labelY, 10, height-60);
            g2d.drawImage(bufferedImageLabel, 21, 31, 30, height - 30,
            0,0, bufferedImageLabel.getWidth(),
            bufferedImageLabel.getHeight(), null);
            for(int c=0; c< labelsY.size(); c++){
                int y =labelY + (labelsY.get(c)*(height-
                60)/bufferedImageLabel.getHeight());
                g2d.drawString((c+1)+"" , labelX-9, y+10);
                g2d.drawLine(labelX, y, labelX+10, y);
            }
        }
    }
}
```

E. MaSOM/src/Gradient.java

```
import java.awt.Color;

/**
 * <p>There are a number of defined gradient types (look at the static
 * fields), but you can create any gradient you like by using either of the
 * following functions:<ul>
 * <li>public static Color[] createMultiGradient(Color[] colors, int
 * numSteps)</li> <li>public static Color[] createGradient(Color one, Color
 * two, int numSteps)</li></ul>
 * You can then assign an arbitrary Color[] object to the HeatMap as follows:
 * <pre>myHeatMap.updateGradient(Gradient.createMultiGradient(new Color[]
 * {Color.red, Color.white, Color.blue}, 256));</pre></p> <hr />
 * <p><strong>Copyright:</strong> Copyright (c) 2007, 2008</p>
 * <p>HeatMap is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.</p>
 * <p>HeatMap is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.</p>
 * <p>You should have received a copy of the GNU General Public License
 * along with HeatMap; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
 * USA</p> @author Matthew Beckler (matthew@mbeckler.org)
 * @author Josh Hayes-Sheen (grey@grevian.org), Converted to use
 * BufferedImage. @author J. Keller (jpaulkeller@gmail.com), Added
 * transparency (alpha) support, data ordering bug fix. @version 1.6
 */

public class Gradient{
       /**
       * Produces a gradient using the University of Minnesota's school colors, * from
       maroon (low) to gold (high)
       */
       public final static Color[] GRADIENT_MAROON_TO_GOLD = createGradient (new
       Color(0xA0, 0x00, 0x00), new Color(0xFF, 0xFF, 0x00), 500);
       /**
       * Produces a gradient from blue (low) to red (high)
       */
       public final static Color[] GRADIENT_BLUE_TO_RED = createGradient(Color.BLUE,
       Color.RED, 500);
       /**
       * Produces a gradient from black (low) to white (high)
       */
       public final static Color[] GRADIENT_BLACK_TO_WHITE = createGradient
       (Color.BLACK, Color.WHITE, 500);
       /**
       *Produces a gradient from red (low) to green (high)
       */
       public final static Color[] GRADIENT_RED_TO_GREEN = createGradient (Color.RED,
       Color.GREEN, 500);
       /**
       *Produces a gradient through green, yellow, orange, red
       */
       public final static Color[] GRADIENT_GREEN_YELLOW_ORANGE_RED =
       createMultiGradient(new Color[]{Color.green, Color.yellow, Color.orange,
       Color.red}, 500);
       /**
       *Produces a gradient through the rainbow: iolet, blue, green, yellow, orange,
       red
```

```
  */
  public final static Color[] GRADIENT_RAINBOW = createMultiGradient(new
  Color[]{new Color(181, 32, 255), Color.blue, Color.green, Color.yellow,
  Color.orange, Color.red}, 500);
  /**
  *Produces a gradient for hot things (black, red, orange, yellow, white)
  */
  public final static Color[] GRADIENT_HOT = createMultiGradient(new
  Color[]{Color.black, new Color(87, 0, 0), Color.red, Color.orange,
  Color.yellow, Color.white}, 500);
  /**
  *Produces a different gradient for hot things (black, brown, orange, white)
  */
  public final static Color[] GRADIENT_HEAT = createMultiGradient(new
  Color[]{Color.black, new Color(105, 0, 0), new Color(192, 23, 0), new
  Color(255, 150, 38), Color.white}, 500);
  /**
  *Produces a gradient through red, orange, yellow
  */
  public final static Color[] GRADIENT_ROY = createMultiGradient(new
  Color[]{Color.red, Color.orange, Color.yellow}, 500);
  /**
  * Creates an array of Color objects for use as a gradient, using a
  * linear interpolation between the two specified colors.
  * @param one Color used for the bottom of the gradient
  * @param two Color used for the top of the gradient
  * @param numSteps The number of steps in the gradient.250 is a good number.
  */
  public static Color[] createGradient(final Color one, final Color two, final
  int numSteps){
     int r1 = one.getRed();
    int g1 = one.getGreen();
    int b1 = one.getBlue();
    int a1 = one.getAlpha();
    int r2 = two.getRed();
    int g2 = two.getGreen();
    int b2 = two.getBlue();
    int a2 = two.getAlpha();
    int newR = 0;
    int newG = 0;
    int newB = 0;
    int newA = 0;

    Color[] gradient = new Color[numSteps];
    double iNorm;
    for (int i = 0; i < numSteps; i++){
        iNorm = i / (double)numSteps; //a normalized [0:1] variable
        newR = (int) (r1 + iNorm * (r2 - r1));
        newG = (int) (g1 + iNorm * (g2 - g1));
        newB = (int) (b1 + iNorm * (b2 - b1));
        newA = (int) (a1 + iNorm * (a2 - a1));
        gradient[i] = new Color(newR, newG, newB, newA);
    }
    return gradient;
}

/**
  * Creates an array of Color objects for use as a gradient, using an array of
    Color objects. It uses a linear interpolation between each pair of points. The
    parameter numSteps defines the total number of colors in the returned array,
    not the number of colors per segment.
  * @param colors An array of Color objects used for the gradient. The Color at
    index 0 will be the lowest color.
```

```
      * @param numSteps The number of steps in the gradient. 250 is a good number.
     */
    public static Color[] createMultiGradient(Color[] colors, int numSteps){
        //we assume a linear gradient, with equal spacing between colors
         //The final gradient will be made up of n 'sections',
         //where n = colors.length - 1
        int numSections = colors.length - 1;
        int gradientIndex = 0;
         //points to the next open spot in the final gradient
        Color[] gradient = new Color[numSteps];
        Color[] temp;

        if (numSections <= 0){
              throw new IllegalArgumentException("You must pass in at least 2 colors in
              the array!");
        }
        for (int section = 0; section < numSections; section++){
              //we divide the gradient into (n - 1) sections, and do a regular gradient
              for each
              temp = createGradient(colors[section], colors[section+1], numSteps /
              numSections);
             for (int i = 0; i < temp.length; i++){
                 //copy the sub-gradient into the overall gradient
                 gradient[gradientIndex++] = temp[i];
             }
        }
        if (gradientIndex < numSteps){
             //The rounding didn't work out in our favor, and there is at
             //least one unfilled slot in the gradient[] array.
             //We can just copy the final color there
              for (/* nothing to initialize */; gradientIndex < numSteps;
              gradientIndex++){
                 gradient[gradientIndex] = colors[colors.length - 1];
             }
        }

        return gradient;
    }
}
```

F. MaSOM/src/Preprocess.java

```
import java.util.Arrays;
import java.util.Comparator;

public class Preprocess {
       public double data[][],rawData[][];
       private int rows, cols;
       public boolean isLogTransformed, isZScoreTransformed, isScaleNormalized,
       isMedianCentered,isQuantileNormalized;

       public Preprocess(double data[][]){
              this.rawData = data;
              this.data = new double[data.length][data[0].length];
              for (int iter = 0; iter < data.length; iter++)
                     this.data[iter] = data[iter].clone();
              rows = data.length;
              cols = data[0].length;
              isLogTransformed = false;
              isZScoreTransformed = false;
```

```
        isScaleNormalized = false;
        isMedianCentered = false;
        isQuantileNormalized = false;
    }

    public double[][] getPreprocessedData(){
        return data;
    }
    public void resetPreprocessedData(){
        this.data = new double[rawData.length][rawData[0].length];
        for (int iter = 0; iter < rawData.length; iter++)
            this.data[iter] = rawData[iter].clone();
    }
    public void logTransform(){
        double temp = 1 / Math.log(2);  // base
        for(int i=0; i< rows; i++)
            for(int j =0; j< cols; j++){
                data[i][j]= Math.log(data[i][j]) * temp;
                if (data[i][j] == Double.NEGATIVE_INFINITY)
                data[i][j] = 0;
            }
        isLogTransformed = true;
    }

    public void zScoreTransform(){
        for(int i=0; i<rows; i++){ // for every row
            double sum = 0;
            for(int j=0; j<cols; j++)// compute for mean
                sum += data[i][j];
            double mean = sum/cols;
            sum = 0;//compute for standard deviation
            for(int j=0; j<cols; j++)
                sum += Math.pow(data[i][j] - mean, 2);
            double std = Math.sqrt( sum/(cols-1) );
            for(int j=0; j<cols; j++){// transform
                if(std == 0) data[i][j] = 0;
                else data[i][j] = (data[i][j] - mean)/std;
            }
        }
        isZScoreTransformed = true;
    }

    public double getColMedian(int j){
        double[] invCol = new double[rows];
        for(int i=0; i<rows; i++)
            invCol[i] = data[i][j];
        Arrays.sort(invCol);
        if(rows%2==1) return invCol[rows/2]; //odd
        else return (invCol[rows/2] + invCol[(rows/2)-1])/2; //even
    }

    public double getColMAD(int j){
        double[] invCol = new double[rows];
        for(int i=0; i<rows; i++)
            invCol[i] = Math.abs(data[i][j]);
        Arrays.sort(invCol);
        if(rows%2==1) return invCol[rows/2]; //odd
        else return (invCol[rows/2] + invCol[(rows/2)-1])/2; //even
    }

    public void medianCenter(){
        for(int j =0; j<cols; j++){
            double median = getColMedian(j);
```

```
                for(int i=0; i< rows; i++)
                        data[i][j] -= median;
        }
        isMedianCentered = true;
}

public void scaleNormalize(){
        if(!isMedianCentered){
                if(!isLogTransformed){
                        if(isZScoreTransformed)    resetPreprocessedData();
                        logTransform();
                        isLogTransformed = false;
                        if(isZScoreTransformed)    zScoreTransform();
                }
                medianCenter();
                isMedianCentered = false;
        }else if(!isLogTransformed){
                resetPreprocessedData();
                logTransform();
                isLogTransformed = false;
                if(isZScoreTransformed) zScoreTransform();
                medianCenter();
        }

        double MAD[] = new double[cols];
        double c = 1; // c = product of all MAD[j]
        for(int j=0; j<cols; j++){
                MAD[j] = getColMAD(j);
                c *= MAD[j];
        }
        c = Math.pow(c, (double) 1/cols); // jth root
        for(int j=0; j<cols; j++){
                double temp = c/MAD[j]; //scale factor
                for(int i=0; i<rows; i++)
                        data[i][j] *= temp;
        }
        isScaleNormalized = true;
}

private class dataWithIndex{
        double val;
        int ind; // keeps track of the actual row index of the cell
        public dataWithIndex(double data, int index){
                this.val = data;
                this.ind = index;
        }
}

public void quantileNormalize(){
        System.setProperty("java.util.Arrays.useLegacyMergeSort", "true");
        Comparator<dataWithIndex> d = new Comparator<dataWithIndex>(){
                public int compare(dataWithIndex d1, dataWithIndex d2){
                        return Double.compare(d1.val, d2.val);
                }
        };
        dataWithIndex invRanking[][] = new dataWithIndex[cols][rows];
        for(int j=0; j<cols; j++){
        // rank the elements of each column in ascending order
                for(int i=0; i<rows; i++){
                        invRanking[j][i] = new dataWithIndex(data[i][j], i);
                }
                Arrays.sort(invRanking[j], d);
        }
```

```
              for(int i=0; i<rows; i++){
                     double ave=0; //get the average of numbers with same rank
                     for(int j=0; j<cols; j++)
                            ave+=data[invRanking[j][i].ind ][j];//use the ranking
                     ave /= cols;
                     // replace all elements of the same rank with their average
                     for(int j=0; j<cols; j++)
                            data[ invRanking[j][i].ind ][j] = ave;
              }
              isQuantileNormalized = true;
       }
}
```

G. MaSOM/src/FileParser.java

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.NoSuchElementException;
import java.util.StringTokenizer;

/**
* This class reads and parses files and stores the contents to its data
structures.
 */
public class FileParser{
       private String dataPath, rowsPath,colsPath; // location of data
       public double data[][]; // data file
       public int dataNumOfRows, dataNumOfCols; // dimension of numerical data
       public String dataRowLabels[], dataColLabels[]; // identification keys
       public int rowNumOfDescriptions, colNumOfDescriptions;
       // number of categories, excluding the id
       public String dataRowDescription[][], dataColDescription[][];
       // 1st row contains the label of the categories

       public FileParser(){
              dataRowDescription = null;
              dataColDescription = null;
       }

       public void setDataFile(String path){
              dataPath = path;
       }
       public void setGeneFile(String path){
              rowsPath = path;
       }
       public void setSampleFile(String path){
              colsPath = path;
       }
       /**
        * Parses the expression ratio file.
        * @return True if parsing is successful, otherwise False
        */
       public boolean extractDataContents(){
              try {
                     BufferedReader read = new BufferedReader(new
                     FileReader(dataPath));
                     String line = "";
                     int fileRowCount=0;// count the number of rows of the file
                     for(fileRowCount=0;read.ready();fileRowCount++)
                         read.readLine();
                     if (fileRowCount <= 1) return false;
                     read = new BufferedReader(new FileReader(dataPath));
                     // re-read the file
                     if(read.ready()){
                         line = read.readLine();
                         // first line of file, contains Column IDs
                         StringTokenizer token=new StringTokenizer(line,"\t");
                         dataNumOfCols = token.countTokens() - 1 ;
                         // all columns except the 1st
                         if(dataNumOfCols<=1) return false;
                         dataColLabels = new String[dataNumOfCols];
                         // excluding the label for the Row IDs
                         token.nextToken(); // label of rowIDs
                         for(int i = 0; i<dataNumOfCols ; i++)
                             dataColLabels[i] = token.nextToken();
                     }

                     dataNumOfRows = fileRowCount - 1 ;
                     // all rows except the 1st
                     dataRowLabels = new String[dataNumOfRows];
                     // excluding 1st row, already in dataColLabels[0]
                     data = new double[dataNumOfRows][dataNumOfCols];
                     for(int i = 0; i<dataNumOfRows; i++){
                         line = read.readLine();
                         StringTokenizer token=new StringTokenizer(line,"\t");
                         dataRowLabels[i] = token.nextToken();
                         for(int j = 0; j<dataNumOfCols; j++){
                             try{
                                 data[i][j]=Double.parseDouble
                                 (token.nextToken());
                             }catch(NoSuchElementException e){return false;}
                             catch(NumberFormatException e){return false;}
                         }
                     }
                     read.close();
              }
              catch (NegativeArraySizeException e) {return false;}
              catch (FileNotFoundException e) {return false;}
              catch (IOException e) {return false;}
              return true;
       }

       public boolean extractDescriptionContents(boolean isRows){
              try {
                     BufferedReader read;
                     String line = "";
                     if(isRows){ // extract gene or probe description
                         read = new BufferedReader(new FileReader(rowsPath));
                         for(int i = 0; i<dataNumOfRows +1; i++){
                             line = read.readLine();
                             String[] temp = line.split("\\t");
                             if(i==0){
                                 rowNumOfDescriptions = temp.length -1 ;
                                 //exclude the row ID from the count
                                 dataRowDescription = new
                                 String[dataNumOfRows+1]
                                 [rowNumOfDescriptions+1];
                             }
                             for(int j = 0; j<rowNumOfDescriptions+1; j++){
                                 try{
                                     if(temp[j].length() == 0)
                                       dataRowDescription[i][j]="-";
                                     else
                                       dataRowDescription[i][j]=temp[j];
                                 }catch(IndexOutOfBoundsException e){
                                     dataRowDescription[i][j] ="-";
                                 }
                             }
                         }
                     }
                     else{ // extract sample description
                         read = new BufferedReader(new FileReader(colsPath));
                         for(int i = 0; i<dataNumOfCols +1; i++){
                             line = read.readLine();
                             String[] temp = line.split("\\t");
                             if(i==0){
                                 colNumOfDescriptions = temp.length -1 ;
                                 //exclude the column ID from the count
                                 dataColDescription = new String[dataNumOfCols
                                 + 1 ] [colNumOfDescriptions + 1 ];
                             }
                             for(int j = 0; j<colNumOfDescriptions+1; j++){
                                 try{
                                     if(temp[j].length() == 0)
                                       dataColDescription[i][j]="-";
                                     else
                                       dataColDescription[i][j]=temp[j];
                                 }catch(IndexOutOfBoundsException e){
                                     dataColDescription[i][j] ="-";
                                 }
                             }
                         }
                     }
                     if(read.read()!=-1) return false;
                     read.close();
              }
              catch (NoSuchElementException e){ return false;}
              catch (NullPointerException e){ return false;}
              catch (FileNotFoundException e) {return false;}
              catch (IOException e) {return false;}
              return true;
       }
   }
```

H. MaSOM/src/FileExporter.java

```
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.Toolkit;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import javax.imageio.ImageIO;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.table.TableModel;
import com.itextpdf.text.Document;
import com.itextpdf.text.Font;
import com.itextpdf.text.Image;
import com.itextpdf.text.PageSize;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.Rectangle;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;


public class FileExporter {
	public boolean isFilenameValid(String file) {
		File f = new File(file);
		try {
			f.getCanonicalPath();
			return true;
		}
		catch (IOException e) {
			return false;
		}
	}

	boolean canWriteToDirectory(String path) {
		File file=new File(path);
		if (file.exists())
			return file.canWrite();
		else {
			try {
				file.createNewFile();
				file.delete();
				return true;
			}
			catch (Exception e) {return false;	}
		}
	}

	public boolean saveHeatmapPanelToImage(JPanel panel,String filename){
		try {
			File outputfile = new File(filename);
			ImageIO.write( convertPanelToImage(panel,500),"png", outputfile);
		} catch (Exception e) {return false;}
		return true;
	}

	public BufferedImage convertPanelToImage(JPanel panel, double width){
		BufferedImage originalImage = new BufferedImage(panel.getWidth(),
			panel.getHeight(),BufferedImage.TYPE_BYTE_INDEXED);
		Graphics2D gg = originalImage.createGraphics();
		gg.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
			RenderingHints.VALUE_INTERPOLATION_BILINEAR);
		gg.setRenderingHint(RenderingHints.KEY_RENDERING,
			RenderingHints.VALUE_RENDER_QUALITY);
		gg.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
			RenderingHints.VALUE_ANTIALIAS_ON);
		panel.paint(gg);
		gg.dispose();

		double s = width/originalImage.getWidth();
		BufferedImage after = new BufferedImage((int) width,
		(int)(s*originalImage.getHeight()) , BufferedImage.TYPE_INT_ARGB);
		AffineTransform at = new AffineTransform();
		at.scale(s, s);
```

```
        AffineTransformOp scaleOp = new AffineTransformOp(at,
        AffineTransformOp.TYPE_BILINEAR);
        after = scaleOp.filter(originalImage, after);
        return after;
}
public boolean exportTablesToFile(ArrayList <JTable> table, String filename) {
        File file = new File(filename);
        try {
                FileWriter out = new FileWriter(file);
                for(int t=0; t<table.size(); t++){
                        TableModel model = table.get(t).getModel();
                        for(int i=0; i < model.getColumnCount();i++)
                                out.write(model.getColumnName(i)+"\t");
                        out.write("\n");
                        for(int i=0; i < model.getRowCount();i++){
                                for(int j=0;j < model.getColumnCount();j++)
                                        out.write(model.getValueAt(i, j)
                                        .toString()+"\t");
                                out.write("\n");
                        }
                        out.write("\n");
                }
                out.close();
        } catch (Exception e) {return false;}
        return true;
}

public boolean exportToPDF(ArrayList <JTable> table, String filename, JPanel
heatmapPanel){
        Document document=new Document();
        Font headerFont=new Font(), contentFont = new Font();
        headerFont.setSize(10);
        headerFont.setStyle(Font.BOLD);
        contentFont.setSize(8);
        PdfPTable tab;
        try{
                PdfWriter.getInstance(document,new FileOutputStream (filename));
                document.open();
                if(heatmapPanel !=null){
                        BufferedImage bi = convertPanelToImage(heatmapPanel,500);
                        Image itextImage = Image.getInstance(Toolkit.get
                        DefaultToolkit().createImage(bi.getSource()), null);
                        document.setPageSize(new Rectangle(595,100 +
                        itextImage.getHeight()));
                        document.newPage();
                        document.add(itextImage);
                }
                //tables
                document.setPageSize(PageSize.A4);
                document.newPage();
                for(int t=0; t<table.size(); t++){
                        int cols =table.get(t).getColumnCount();
                        int rows = table.get(t).getRowCount();
                        tab=new PdfPTable(cols);
                        for(int i=0; i < cols;i++)
                                tab.addCell(new Phrase (""+table.get(t)
                                .getColumnName(i),headerFont));
                        for(int i=0;i<rows;i++)
                                for(int j=0;j<cols;j++)
                                        tab.addCell(new Phrase(""+table.get(t).
                                        getModel().getValueAt(i,j),contentFont));
                        document.add(tab);
                        document.add( new Paragraph( "\n" ) );
```

```
            }
            document.close();
        }catch (Exception e) {return false;}
        return true;
    }
    public boolean exportClusterDataToFile( double[][] data,int[]
    clusterAssignments, boolean clusterSamples, String[] rowLabels, String[]
    colLabels,int cluster, String filename) {
        File file = new File(filename);
        try {
            FileWriter out = new FileWriter(file);
            out.write("Cluster "+ (cluster+1)+"\t" );
            if(clusterSamples){
                for(int j=0;j < data.length;j++)
                    if(clusterAssignments[j] == cluster)
                        out.write(colLabels[j]+"\t");
                for(int j=0;j < data[0].length;j++){
                    out.write("\n" + rowLabels[j] +"\t");
                    for(int i=0; i < data.length;i++)
                        if(clusterAssignments[i] == cluster)
                            out.write(data[i][j]+"\t");
                }
            }
            else{
                for(int j=0;j < data[0].length;j++)
                    out.write(colLabels[j]+"\t");
                for(int i=0; i < data.length;i++){
                    if(clusterAssignments[i] == cluster){
                        out.write("\n" + rowLabels[i] +"\t");
                        for(int j=0;j < data[0].length;j++)
                            out.write(data[i][j]+"\t");
                    }
                }
            }
            out.close();
        } catch (Exception e) {return false;}
        return true;
    }
    public boolean exportClusterDescToFile( String[][] data,int[]
    clusterAssignments,int cluster, String filename) {
        File file = new File(filename);
        try {
            FileWriter out = new FileWriter(file);
            for(int i=0; i < data.length;i++){
                if(i==0 || clusterAssignments[i-1] == cluster){
                    for(int j=0;j < data[0].length;j++)
                        out.write(data[i][j]+"\t");
                    out.write("\n");
                }
            }
            out.close();
        } catch (Exception e) {return false;}
        return true;
    }
}
```

I. MaSOM/src/ResultingData.java

```java
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Arrays;

public class ResultingData{
	Object paramUsedTableContents[][];
	Object clustPropertiesTableContents[][];
	Object expRatioTableContents[][];
	Object expRatioTableCols[];
	ResultsGUI rgui;
	double processedData[][],centroids[][], rawData[][];
	double clustVariances[], clustMSA[],clustMSW[];
	boolean log, med, quant, scale, zScore, clusterSamples;
	double initLearningRate, overAllVariance;
	int desiredClusters, iterations, distMetric, numOfFinalClusters;
	int clusterAssignment[], clustSizes[];
	String geneIDs[],sampleIDs[],geneDescription[][],sampleDescription[][];
	javax.swing.JPanel heatmap;

	public ResultingData(javax.swing.JPanel heatmap,double data[][]){
		this.heatmap = heatmap;
		this.processedData=data;
	}

	public void showWindow(){
		rgui = new ResultsGUI(paramUsedTableContents,
		""+numOfFinalClusters,""+overAllVariance,clustPropertiesTableContents,exp
		RatioTableContents,expRatioTableCols);
		rgui.setHeatmap(heatmap);
		rgui.setSelectiveExportTableModel(processedData,clusterAssignment,cluster
		Samples,geneIDs,sampleIDs,(clusterSamples? sampleDescription :
		geneDescription ));
		rgui.initComponents();
		rgui.setVisible(true);
	}

	public void getPreprocMethodsUsed(boolean log, boolean med, boolean
	quant,boolean scale,boolean zScore){
		this.log= log;
		this.med= med;
		this.quant= quant;
		this.scale= scale;
		this.zScore = zScore;
	}
	public void getClusteringParameters(int clusters, boolean isByCols, double
	initLearningRate, int maxIter, int distMetric){
		this.desiredClusters = clusters;
		this.clusterSamples = isByCols;
		this.initLearningRate = initLearningRate;
		this.iterations = maxIter;
		this.distMetric = distMetric;
	}
	public void generateParamUsed(){
		String preprocList ="";
		if(log) preprocList = "Logarithmic Transformation";
		if(zScore){
			if(!preprocList.equals(""))preprocList+=", ";
			preprocList += "Z-Score Transformation";
		}
		if(med){
			if(!preprocList.equals(""))preprocList+=", ";
```

```
            preprocList += "Median Centering";
        }
        if(scale){
            if(!preprocList.equals(""))preprocList+=", ";
            preprocList += "Scale Normalization";
        }
        if(quant){
            if(!preprocList.equals(""))preprocList+=", ";
            preprocList += "Quantile Normalization";
        }
        if(preprocList.equals(""))preprocList="None";

        String distMetricStr = "Manhattan";
        if(distMetric==2)distMetricStr = "Euclidian";
        else if(distMetric == 3)distMetricStr = "Supnorm";
        Object[][] temp ={
                {"Number of genes", geneIDs.length},
                {"Number of samples",sampleIDs.length},
                {"Preprocessing methods",preprocList},
                {"Clustering",(clusterSamples? "Samples":"Genes")},
                {"Initial number of clusters",desiredClusters},

                {"Distance metric",distMetricStr},
                {"Initial learning rate",initLearningRate},
                {"Number of iterations",iterations},
        };
        paramUsedTableContents = temp;
    }

    public void getClusters(double centroids[][], int numOfFinalClusters){
        this.centroids = centroids;
        this.numOfFinalClusters = numOfFinalClusters;
    }

    public void getOverAllMSD(){
        double sumOfSquares =0;
        int d = 0;
        for(int i = 0; i< centroids.length; i++)
            for(int j = i+1; j< centroids.length; j++){
                if(clustSizes[i]>0 && clustSizes[j]>0){
                    d++;
                    for(int k = 0; k< centroids[0].length; k++)
                        sumOfSquares += Math.pow(centroids[i][k]-
                        centroids[j][k], 2);
                }
            }
        if(d == 0) overAllVariance =0;
        else overAllVariance = sumOfSquares/d;
    }

    public double getMSW(double x[][], double y[]){
        double sum = 0;
        for(int j=0; j<x.length; j++){
            sum += getSSW(x[j], y);
        }
        return sum/(x.length-1);
    }

    public double getSSW(double x[], double y[]){
        double SSW =0;
        for(int i=0; i< x.length; i++)
            SSW += (x[i] - y[i]) * (x[i] - y[i]);
        return SSW;
```

```java
}

public double getMSA(int c){
        double sum =0;
        for(int i=0; i<centroids.length; i++){
                if(clustSizes[i]>0)
                        for(int j=0; j<centroids[0].length; j++)
                                sum += (centroids[c][j]-centroids[i][j])*
                                (centroids[c][j]-centroids[i][j]);
        }
        return sum/(centroids.length);
}

public void getClusterAssignments(int clusterAssignment[]){
        this.clusterAssignment = clusterAssignment;
}

@SuppressWarnings("unchecked")
public void getClusterVariances(){
        @SuppressWarnings("rawtypes")
        ArrayList temp[] = new ArrayList[centroids.length];
        clustSizes = new int[centroids.length];
        for(int c=0; c < centroids.length; c++){
                clustSizes[c] = 0;
                temp[c] = new ArrayList<>();
                for(int d=0; d < processedData.length; d++){
                        if(clusterAssignment[d]==c){
                                clustSizes[c]++;
                                temp[c].add(processedData[d]);
                        }
                }
        }
        clustVariances = new double[centroids.length];
        clustMSW = new double[centroids.length];
        clustMSA = new double[centroids.length];
        for(int c=0; c < centroids.length; c++){
                int size = temp[c].size();
                double arr[][] = new double[size][centroids[0].length];
                for(int j = 0; j < size; j++) {
                        arr[j] = (double[]) temp[c].get(j);
                }
                if(clustSizes[c]>0){
                        clustMSW[c] = getMSW(arr,centroids[c]);
                        clustMSA[c] = getMSA(c);
                        if(clustMSW[c] == 0) clustVariances[c] = clustMSA[c];
                        else clustVariances[c] = clustMSA[c]/clustMSW[c];
                }
        }
}

@SuppressWarnings("unchecked")
public void generateClustProperties(){
        @SuppressWarnings("rawtypes")
        ArrayList tempList = new ArrayList();
        DecimalFormat df = new DecimalFormat("#.##");
        for(int i=0; i<centroids.length; i++)
                if(clustSizes[i]>0) tempList.add(i);
        clustPropertiesTableContents = new Object[tempList.size()][4];
        for(int i=0; i<tempList.size(); i++){
                int index = (int) tempList.get(i);
                clustPropertiesTableContents[i][0] = i+1;
                clustPropertiesTableContents[i][1] = clustSizes[index];
                try{
```

```
            clustPropertiesTableContents[i][2] = Double.
            parseDouble(df.format(clustVariances[index]));
        }catch(NumberFormatException  e)
        {clustPropertiesTableContents[i][2] = 0.0;}
        Double tempObj[] = new Double[centroids[0].length];
        for(int j=0; j<centroids[0].length; j++)
            tempObj[j] = Double.parseDouble
            (df.format(centroids[index][j]));
        clustPropertiesTableContents[i][3] = Arrays.
        deepToString(tempObj);
    }
}

public void getOriginalData( double data[][],String dataRowLabels[], String
dataColLabels[]){
    this.rawData = data;
    this.geneIDs =dataRowLabels;
    this.sampleIDs = dataColLabels;
}
public void getDataDescriptions(String geneDescription[][], String
sampleDescription[][]){
    this.geneDescription = geneDescription;
    this.sampleDescription = sampleDescription;
}

public void generateExpRatio(){
    int cols= 2;
    if(clusterSamples){
        if(sampleDescription!=null)
            cols+= sampleDescription[0].length -1;
    }else if(geneDescription!=null)
        cols+= geneDescription[0].length -1;

    expRatioTableCols = new Object[cols];
    expRatioTableCols[0]="Cluster Number";
    expRatioTableCols[1]=(clusterSamples? "Sample" : "Gene") + " ID";
    if(clusterSamples && sampleDescription!=null)
        for(int j=1; j<sampleDescription[0].length; j++)
            expRatioTableCols[j+1] = sampleDescription[0][j];
    else if(!clusterSamples && geneDescription!=null)
        for(int j=1; j<geneDescription[0].length; j++)
            expRatioTableCols[j+1] = geneDescription[0][j];
    int tempClustAssign[] = new int[clusterAssignment.length];
    int clusterCount = 0;
    boolean isEmpty;
    if(clusterSamples)
        for(int clust = 0; clust<desiredClusters; clust++){
            isEmpty = true;
            for(int i = 0; i<rawData[0].length; i++){
                if(clusterAssignment[i]==clust){
                    isEmpty = false;
                    tempClustAssign[i] = clusterCount;
                }
            }if(!isEmpty) clusterCount++;
        }
    else
        for(int clust = 0; clust<desiredClusters; clust++){
            isEmpty = true;
            for(int i = 0; i<rawData.length; i++){
                if(clusterAssignment[i]==clust){
                    isEmpty = false;
                    tempClustAssign[i] = clusterCount;
                }
                }if(!isEmpty) clusterCount++;
            }
        clusterAssignment = tempClustAssign;
        int rows= (clusterSamples? rawData[0].length : rawData.length);
        expRatioTableContents = new Object[rows][cols];
        for(int i=0; i<rows; i++){
            expRatioTableContents[i][0] = clusterAssignment[i]+1;
            expRatioTableContents[i][1] = (clusterSamples? sampleIDs[i]:
            geneIDs[i]);
            if(clusterSamples && sampleDescription!=null)
                for(int j=1; j<sampleDescription[0].length; j++)
                    expRatioTableContents[i][j+1] =
                    sampleDescription[i+1][j];
            else if(!clusterSamples && geneDescription!=null)
                for(int j=1; j<geneDescription[0].length; j++)
                    expRatioTableContents[i][j+1] =
                    geneDescription[i+1][j];
        }
    }
}
```

J. MaSOM/src/ResultsGUI.java

```
import java.awt.Cursor;
import java.awt.Toolkit;
import java.net.URL;
import java.util.ArrayList;
import java.util.concurrent.CancellationException;
import java.util.concurrent.ExecutionException;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.SwingWorker;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableCellRenderer;

@SuppressWarnings("serial")
public class ResultsGUI extends javax.swing.JFrame {
    public Object paramUsedTableContents[][],clustProperties  TableContents[][],
    expRatioTableContents[][],expRatioTableCols[];
    public String numOfClustStr,overAllVariance;
    public JTableModel tableWithExports;
    private exportFileWorker exportFile;
    public javax.swing.JPanel heatmap;
    private javax.swing.JScrollPane scroll;
    private javax.swing.JFileChooser folderChooser = new JFileChooser();
    public javax.swing.JTable clustPropertiesTableWithOutButtons;
    private javax.swing.JButton chooseDestFolderButton;
    private javax.swing.JPanel clustPropertiesPanel;
    private javax.swing.JScrollPane clustPropertiesScrollPane;
    private javax.swing.JTable clustPropertiesTable;
    private javax.swing.JCheckBox contentPDFClustersCheckBox;
    private javax.swing.JCheckBox contentPDFExpRatioCheckBox;
    private javax.swing.JCheckBox contentPDFHeatMapCheckBox;
    private javax.swing.JCheckBox contentPDFParametersUsedCheckBox;
    private javax.swing.JCheckBox contentPNGHeatMapCheckBox;
    private javax.swing.JLabel contentPNGHeatmapLabel;
    private javax.swing.JCheckBox contentTXTClustersCheckBox;
    private javax.swing.JCheckBox contentTXTExpRatioCheckBox;
    private javax.swing.JCheckBox contentTXTParametersUsedCheckBox;
    private javax.swing.JCheckBox contentXLSClustersCheckBox;
    private javax.swing.JCheckBox contentXLSExpRatioCheckBox;
    private javax.swing.JLabel contentsLabel;
    private javax.swing.JLabel destFolderLabel;
    private javax.swing.JTextField destFolderPath;
    private javax.swing.JLabel errorMessageForm4;
    private javax.swing.JPanel expRatioPanel;
    private javax.swing.JScrollPane expRatioScrollPane;
    public javax.swing.JTable expRatioTable;
    private javax.swing.JButton exportButton;
    private javax.swing.JTextField exportExcelFileName;
    private javax.swing.JTextField exportImageFileName;
    private javax.swing.JTextField exportPDFFileName;
    private javax.swing.JTextField exportTextFileName;
    private javax.swing.JCheckBox exportToExcelCheckbox;
    private javax.swing.JCheckBox exportToImageCheckbox;
    private javax.swing.JCheckBox exportToPDFCheckbox;
    private javax.swing.JCheckBox exportToTextCheckbox;
    private javax.swing.JLabel fileFormatsLabel;
    private javax.swing.JLabel fileNamesLabel;
    private javax.swing.Box.Filler filler1;
    private javax.swing.JPanel form4;
    private javax.swing.JLabel numOfClustResultLabel;
    private javax.swing.JLabel numOfClustResultText;
    private javax.swing.JLabel overAllVarResultLabel;
    private javax.swing.JLabel overAllVarResultText;
    private javax.swing.JPanel paramUsedPanel;
    private javax.swing.JScrollPane paramUsedScrollPane;
    public javax.swing.JTable paramUsedTable;
    private javax.swing.JLabel pdfLabel;
    private javax.swing.JLabel pngLabel;
    private javax.swing.JProgressBar progressbar;
    private javax.swing.JButton resetButtonForm4;
    private javax.swing.JLabel status;
    public javax.swing.JTabbedPane tabbedPane;
    private javax.swing.JLabel txtLabel;
    private javax.swing.JLabel xlsLabel;
    private javax.swing.JLabel jLabel1;

    public ResultsGUI(Object[][] paramUsedTableContents, String numOfClustStr,String
    overAllVariance, Object[][] clustPropertiesTableContents, Object[][]
    expRatioTableContents, Object[] expRatioTableCols) {
             this.paramUsedTableContents=paramUsedTableContents;
             this.numOfClustStr = numOfClustStr;
             this.overAllVariance =overAllVariance;
             this.clustPropertiesTableContents = clustPropertiesTableContents;
             this.expRatioTableContents = expRatioTableContents;
             this.expRatioTableCols = expRatioTableCols;
             try { // Set System L&F
                    javax.swing.UIManager.setLookAndFeel(javax.swing.UIManager.getSyst
                    emLookAndFeelClassName());
             } catch (javax.swing.UnsupportedLookAndFeelException |
             ClassNotFoundException | InstantiationException | IllegalAccessException
             e) {}
      }

      public void setHeatmap(javax.swing.JPanel heatmap){
             this.heatmap = heatmap;
             scroll = new javax.swing.JScrollPane(heatmap);
      }
```

```
public void setSelectiveExportTableModel(double[][] data, int[]
clusterAssignment,boolean clusterSamples,String[] geneIDs, String[] sampleIDs,
String[][] description){
       tableWithExports = new JTableModel(
                     new String [] {"Cluster Number", "Number of Members",
                     "Variance","Centroid"},
                     this.clustPropertiesTableContents );
       tableWithExports.setExportData(data,clusterAssignment,clusterSamples,
       geneIDs,sampleIDs, description);

}

public void initComponents() {
   clustPropertiesTableWithOutButtons = new JTable();
   tabbedPane = new javax.swing.JTabbedPane();
  paramUsedPanel = new javax.swing.JPanel();
  paramUsedScrollPane = new javax.swing.JScrollPane();
  paramUsedTable = new javax.swing.JTable();
  clustPropertiesPanel = new javax.swing.JPanel();
  clustPropertiesScrollPane = new javax.swing.JScrollPane();
  clustPropertiesTable = new javax.swing.JTable();
  numOfClustResultLabel = new javax.swing.JLabel();
  numOfClustResultText = new javax.swing.JLabel();
  overAllVarResultLabel = new javax.swing.JLabel();
  overAllVarResultText = new javax.swing.JLabel();
  expRatioPanel = new javax.swing.JPanel();
  expRatioScrollPane = new javax.swing.JScrollPane();
  expRatioTable = new javax.swing.JTable();
  form4 = new javax.swing.JPanel();
  exportButton = new javax.swing.JButton();
  errorMessageForm4 = new javax.swing.JLabel();
  resetButtonForm4 = new javax.swing.JButton();
  chooseDestFolderButton = new javax.swing.JButton();
  destFolderLabel = new javax.swing.JLabel();
  destFolderPath = new javax.swing.JTextField();
  exportToImageCheckbox = new javax.swing.JCheckBox();
  exportToPDFCheckbox = new javax.swing.JCheckBox();
  exportToExcelCheckbox = new javax.swing.JCheckBox();
  exportToTextCheckbox = new javax.swing.JCheckBox();
  fileFormatsLabel = new javax.swing.JLabel();
  contentsLabel = new javax.swing.JLabel();
  fileNamesLabel = new javax.swing.JLabel();
  contentXLSExpRatioCheckBox = new javax.swing.JCheckBox();
  contentXLSClustersCheckBox = new javax.swing.JCheckBox();
  contentTXTExpRatioCheckBox = new javax.swing.JCheckBox();
  contentTXTClustersCheckBox = new javax.swing.JCheckBox();
  contentTXTParametersUsedCheckBox = new javax.swing.JCheckBox();
  contentPDFParametersUsedCheckBox = new javax.swing.JCheckBox();
  contentPDFHeatMapCheckBox = new javax.swing.JCheckBox();
  contentPDFClustersCheckBox = new javax.swing.JCheckBox();
  contentPDFExpRatioCheckBox = new javax.swing.JCheckBox();
  contentPNGHeatMapCheckBox = new javax.swing.JCheckBox();
  exportPDFFileName = new javax.swing.JTextField();
  pdfLabel = new javax.swing.JLabel();
  txtLabel = new javax.swing.JLabel();
  exportTextFileName = new javax.swing.JTextField();
  xlsLabel = new javax.swing.JLabel();
  exportExcelFileName = new javax.swing.JTextField();
  pngLabel = new javax.swing.JLabel();
  exportImageFileName = new javax.swing.JTextField();
  contentPNGHeatmapLabel = new javax.swing.JLabel();
  jLabel1 = new javax.swing.JLabel();
  progressbar = new javax.swing.JProgressBar();
  status = new javax.swing.JLabel();
  filler1 = new javax.swing.Box.Filler(new java.awt.Dimension(0, 0),
new java.awt.Dimension(0, 0), new java.awt.Dimension(0, 32767));
       setTitle("Results");
       URL imgUrl = getClass().getResource("src/img/main.png");
if(imgUrl !=null) super.setIconImage(new ImageIcon(imgUrl).getImage());
else super.setIconImage(new ImageIcon("src/img/main.png").getImage());
       paramUsedTable.setModel(new javax.swing.table.DefaultTableModel(
              paramUsedTableContents,
              new String [] {"Parameter", "Value"     }){
       public boolean isCellEditable(int rowIndex,int columnIndex){
              return false;
       }});

paramUsedTable.getColumnModel().getColumn(0).setPreferredWidth(150);
       paramUsedTable.getColumnModel().getColumn(0).setMaxWidth(150);
       paramUsedTable.setCellSelectionEnabled(true);
       paramUsedTable.getTableHeader().setReorderingAllowed(false);
       paramUsedScrollPane.setViewportView(paramUsedTable);

       javax.swing.GroupLayout paramUsedPanelLayout = new
       javax.swing.GroupLayout(paramUsedPanel);
       paramUsedPanel.setLayout(paramUsedPanelLayout);
       paramUsedPanelLayout.setHorizontalGroup(
       paramUsedPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
       t.LEADING)

.addGroup(paramUsedPanelLayout.createSequentialGroup()
       .addContainerGap().addComponent(paramUsedScrollPane,
       javax.swing.GroupLayout.PREFERRED_SIZE, 761,
       javax.swing.GroupLayout.PREFERRED_SIZE)
       .addContainerGap(37, Short.MAX_VALUE)));
       paramUsedPanelLayout.setVerticalGroup(

paramUsedPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING).addGroup(paramUsedPanelLayout.createSequentialGroup()
.addContainerGap().addComponent(paramUsedScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,Short.MAX_VALUE)));
tabbedPane.addTab("Parameters Used", paramUsedPanel);
clustPropertiesTable = new JTable(tableWithExports);
TableCellRenderer buttonRenderer = new JTableButtonRenderer();
clustPropertiesTable.getColumn("Data").setCellRenderer(buttonRenderer);
clustPropertiesTable.getColumn("Description").setCellRenderer(buttonRenderer);
clustPropertiesTable.addMouseListener(new
JTableButtonMouseListener(clustPropertiesTable));
clustPropertiesTable.setAutoCreateRowSorter(true);
clustPropertiesTableWithOutButtons.setModel(new
javax.swing.table.DefaultTableModel( clustPropertiesTableContents,
new String [] {"Cluster Number", "Number of Members", "Variance", "Centroid"}
       ){
       public boolean isCellEditable(int rowIndex, int columnIndex) {
                     return false;
       }});

DefaultTableCellRenderer alignCenter = new DefaultTableCellRenderer();
alignCenter.setHorizontalAlignment( JLabel.CENTER );
clustPropertiesTable.getColumn("Data").setPreferredWidth(35);
clustPropertiesTable.getColumn("Data").setMaxWidth(35);
       clustPropertiesTable.getColumn("Description").setPreferredWidth(65);
clustPropertiesTable.getColumn("Description").setMaxWidth(65);
clustPropertiesTable.getColumnModel().getColumn(2).setPreferredWidth(90);
clustPropertiesTable.getColumnModel().getColumn(2).setMaxWidth(90);
clustPropertiesTable.getColumnModel().getColumn(2).setCellRenderer( alignCenter
);
clustPropertiesTable.getColumnModel().getColumn(3).setPreferredWidth(110);
clustPropertiesTable.getColumnModel().getColumn(3).setMaxWidth(110);
clustPropertiesTable.getColumnModel().getColumn(3).setCellRenderer( alignCenter
);
clustPropertiesTable.getColumnModel().getColumn(4).setPreferredWidth(80);
clustPropertiesTable.getColumnModel().getColumn(4).setMaxWidth(80);
clustPropertiesTable.getColumnModel().getColumn(4).setCellRenderer(alignCenter)
;
clustPropertiesTable.setCellSelectionEnabled(true);
        clustPropertiesTable.getTableHeader().setReorderingAllowed(false);
clustPropertiesScrollPane.setViewportView(clustPropertiesTable);
numOfClustResultLabel.setText("Number of Clusters:");
numOfClustResultText.setText(numOfClustStr);
overAllVarResultLabel.setText("Mean Squared Distances of Centroids:");
overAllVarResultText.setText(overAllVariance);

javax.swing.GroupLayout clustPropertiesPanelLayout = new
javax.swing.GroupLayout(clustPropertiesPanel);
clustPropertiesPanel.setLayout(clustPropertiesPanelLayout);
clustPropertiesPanelLayout.setHorizontalGroup(clustPropertiesPanelLayout.create
ParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(clustPropertiesPanelLayout.createSequentialGroup()
.addContainerGap().addGroup(clustPropertiesPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
.addComponent(clustPropertiesScrollPane,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 808, Short.MAX_VALUE)
.addGroup(clustPropertiesPanelLayout.createSequentialGroup().addGroup(clustProp
ertiesPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
).addComponent(numOfClustResultLabel)
.addComponent(overAllVarResultLabel)).addGap(18, 18, 18)
.addGroup(clustPropertiesPanelLayout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING).addComponent(overAllVarResultText)
.addComponent(numOfClustResultText)))).addGap(0,37, Short.MAX_VALUE)));
clustPropertiesPanelLayout.setVerticalGroup(clustPropertiesPanelLayout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(clustPropertiesPanelLayout.createSequentialGroup().addContainerGap().
addGroup(clustPropertiesPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.BASELINE).addComponent(numOfClustResultLabel).addComponent(numOfClus
tResultText)).addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D).addGroup(clustPropertiesPanelLayout.createParallelGroup(javax.swing.GroupLay
out.Alignment.BASELINE).addComponent(overAllVarResultLabel).addComponent(overAl
lVarResultText)).addGap(18, 18, 18).addComponent(clustPropertiesScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap(javax.swing.GroupLayout
.DEFAULT_SIZE, Short.MAX_VALUE)));
tabbedPane.addTab("Cluster Properties", clustPropertiesPanel);
tabbedPane.addTab("Heatmap", scroll);
expRatioTable.setAutoCreateRowSorter(true);
        expRatioTable.setModel(new javax.swing.table.DefaultTableModel(
        expRatioTableContents, expRatioTableCols) {
        public boolean isCellEditable(int rowIndex, int columnIndex) {
                return false;
        }});
expRatioTable.getColumnModel().getColumn(0).setPreferredWidth(90);
expRatioTable.getColumnModel().getColumn(0).setMaxWidth(90);
expRatioTable.getColumnModel().getColumn(0).setCellRenderer( alignCenter );
expRatioTable.setDragEnabled(true);
expRatioTable.setCellSelectionEnabled(true);
```

```
expRatioScrollPane.setViewportView(expRatioTable);
javax.swing.GroupLayout expRatioPanelLayout = new
javax.swing.GroupLayout(expRatioPanel);
expRatioPanel.setLayout(expRatioPanelLayout);
expRatioPanelLayout.setHorizontalGroup(
expRatioPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)    .addComponent(expRatioScrollPane,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 808, Short.MAX_VALUE));
expRatioPanelLayout.setVerticalGroup(expRatioPanelLayout.createParallelGroup(ja
vax.swing.GroupLayout.Alignment.LEADING).addComponent(expRatioScrollPane,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 376, Short.MAX_VALUE));

tabbedPane.addTab("Expression Ratios", expRatioPanel);
expRatioTable.setAutoCreateRowSorter(true);
expRatioTable.setModel(new javax.swing.table.DefaultTableModel(
       expRatioTableContents,expRatioTableCols) {
       public boolean isCellEditable(int rowIndex, int columnIndex) {
              return false;
       }});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup().addContainerGap().addGroup(layout.createParallel
Group(javax.swing.GroupLayout.Alignment.TRAILING).addComponent(filler1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE) .addComponent(tabbedPane)).addContainerGap()));
       layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
       .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
       layout.createSequentialGroup() .addComponent(filler1,
javax.swing.GroupLayout.PREFERRED_SIZE, 11,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(tabbedPane, javax.swing.GroupLayout.DEFAULT_SIZE, 404,
Short.MAX_VALUE).addContainerGap()));

form4.setBackground(new java.awt.Color(255, 255, 255));
exportButton.setText("Export");
exportButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
          exportButtonActionPerformed(evt);
      }});
errorMessageForm4.setForeground(new java.awt.Color(255, 0, 0));
errorMessageForm4.setForeground(new java.awt.Color(255, 0, 0));
imgUrl = getClass().getResource("src/img/reset_def.png");
if(imgUrl !=null) resetButtonForm4.setIcon(new ImageIcon(imgUrl));
else resetButtonForm4.setIcon(new
javax.swing.ImageIcon("src/img/reset_def.png"));
resetButtonForm4.setText("Reset");
imgUrl = getClass().getResource("src/img/reset_hov.png");
if(imgUrl !=null) resetButtonForm4.setRolloverIcon(new ImageIcon(imgUrl));
else resetButtonForm4.setRolloverIcon(new
javax.swing.ImageIcon("src/img/reset_hov.png"));
resetButtonForm4.addActionListener(new java.awt.event.ActionListener(){
       public void actionPerformed(java.awt.event.ActionEvent evt) {
              resetButtonForm4ActionPerformed(evt);
       }});
```

```
imgUrl = getClass().getResource("src/img/folder_def.png");
if(imgUrl !=null) chooseDestFolderButton.setIcon(new ImageIcon(imgUrl));
else chooseDestFolderButton.setIcon(new
javax.swing.ImageIcon("src/img/folder_def.png"));
chooseDestFolderButton.setText("Choose");
chooseDestFolderButton.setOpaque(false);
imgUrl = getClass().getResource("src/img/folder_hov.png");
if(imgUrl !=null) chooseDestFolderButton.setRolloverIcon(new
ImageIcon(imgUrl));
else chooseDestFolderButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/folder_hov.png"));
chooseDestFolderButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                chooseDestFolderButtonActionPerformed(evt);
        }});
destFolderLabel.setText("Destination folder");
destFolderPath.setEditable(false);
destFolderPath.setOpaque(false);
exportToImageCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToImageCheckbox.setText("Image");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToImageCheckbox.setIcon(new ImageIcon(imgUrl));
else exportToImageCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToImageCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
if(imgUrl !=null) exportToImageCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToImageCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToImageCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToImageCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToImageCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToImageCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToImageCheckbox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                        exportToImageCheckboxActionPerformed(evt);
        }});
exportToPDFCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToPDFCheckbox.setText("PDF");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToPDFCheckbox.setIcon(new ImageIcon(imgUrl));
        else exportToPDFCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToPDFCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
if(imgUrl !=null) exportToPDFCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToPDFCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToPDFCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToPDFCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToPDFCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToPDFCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToPDFCheckbox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                exportToPDFCheckboxActionPerformed(evt);
        }});
exportToExcelCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToExcelCheckbox.setText("Spreadsheet");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToExcelCheckbox.setIcon(new ImageIcon(imgUrl));
else exportToExcelCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToExcelCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
if(imgUrl !=null) exportToExcelCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToExcelCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToExcelCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToExcelCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToExcelCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToExcelCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToExcelCheckbox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                exportToExcelCheckboxActionPerformed(evt);
        }});
exportToTextCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToTextCheckbox.setText("Text");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToTextCheckbox.setIcon(new ImageIcon(imgUrl));
else exportToTextCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToTextCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
if(imgUrl !=null) exportToTextCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToTextCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToTextCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToTextCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToTextCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToTextCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToTextCheckbox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                exportToTextCheckboxActionPerformed(evt);
        }});
fileFormatsLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
fileFormatsLabel.setText("File Format");
contentsLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
contentsLabel.setText("Contents");
fileNamesLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
fileNamesLabel.setText("File Name");
contentXLSExpRatioCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentXLSExpRatioCheckBox.setText("Expression Ratios");

imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
if(imgUrl !=null){
        contentXLSExpRatioCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
```

```
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentXLSExpRatioCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentXLSExpRatioCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentXLSExpRatioCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentXLSExpRatioCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentXLSExpRatioCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentXLSClustersCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentXLSClustersCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentXLSClustersCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentXLSClustersCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentXLSClustersCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentXLSClustersCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentTXTExpRatioCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentTXTExpRatioCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentTXTExpRatioCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentTXTExpRatioCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentTXTExpRatioCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentTXTExpRatioCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentTXTClustersCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentTXTClustersCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentTXTClustersCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentTXTClustersCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentTXTClustersCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentTXTClustersCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentTXTParametersUsedCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentTXTParametersUsedCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentTXTParametersUsedCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentTXTParametersUsedCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
```

```
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentTXTParametersUsedCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentTXTParametersUsedCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentPDFParametersUsedCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentPDFParametersUsedCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentPDFParametersUsedCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentPDFParametersUsedCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentPDFParametersUsedCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentPDFParametersUsedCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentPDFHeatMapCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentPDFHeatMapCheckBox.setDisabledSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentPDFHeatMapCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentPDFHeatMapCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentPDFHeatMapCheckBox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentPDFHeatMapCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentPDFClustersCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentPDFClustersCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentPDFClustersCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentPDFClustersCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentPDFClustersCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentPDFClustersCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentPDFExpRatioCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentPDFExpRatioCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentPDFExpRatioCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentPDFExpRatioCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentPDFExpRatioCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentPDFExpRatioCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentPNGHeatMapCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentPNGHeatMapCheckBox.setDisabledSelectedIcon(new ImageIcon(imgUrl));
```

```
            imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
            contentPNGHeatMapCheckBox.setIcon(new ImageIcon(imgUrl));
            imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
            contentPNGHeatMapCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
            imgUrl = getClass().getResource("src/img/box_checked_hov.png");
            contentPNGHeatMapCheckBox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
            imgUrl = getClass().getResource("src/img/box_checked_def.png");
            contentPNGHeatMapCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
    }else{
            contentXLSExpRatioCheckBox.setDisabledIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
            contentXLSExpRatioCheckBox.setDisabledSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_dis.png"));
            contentXLSExpRatioCheckBox.setIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
            contentXLSExpRatioCheckBox.setRolloverIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
            contentXLSExpRatioCheckBox.setRolloverSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_hov.png"));
            contentXLSExpRatioCheckBox.setSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_def.png"));
            contentXLSClustersCheckBox.setDisabledIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
            contentXLSClustersCheckBox.setDisabledSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_dis.png"));
            contentXLSClustersCheckBox.setIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
            contentXLSClustersCheckBox.setRolloverIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
            contentXLSClustersCheckBox.setRolloverSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_hov.png"));
            contentXLSClustersCheckBox.setSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_def.png"));
            contentTXTExpRatioCheckBox.setDisabledIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
            contentTXTExpRatioCheckBox.setDisabledSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_dis.png"));
            contentTXTExpRatioCheckBox.setIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
            contentTXTExpRatioCheckBox.setRolloverIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
            contentTXTExpRatioCheckBox.setRolloverSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_hov.png"));
            contentTXTExpRatioCheckBox.setSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_def.png"));
            contentTXTClustersCheckBox.setDisabledIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
            contentTXTClustersCheckBox.setDisabledSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_dis.png"));
            contentTXTClustersCheckBox.setIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
            contentTXTClustersCheckBox.setRolloverIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
            contentTXTClustersCheckBox.setRolloverSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_hov.png"));
            contentTXTClustersCheckBox.setSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_def.png"));
            contentTXTParametersUsedCheckBox.setDisabledIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
            contentTXTParametersUsedCheckBox.setDisabledSelectedIcon(new
            javax.swing.ImageIcon("src/img/box_checked_dis.png"));
            contentTXTParametersUsedCheckBox.setIcon(new
            javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
```

```
contentTXTParametersUsedCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentTXTParametersUsedCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentTXTParametersUsedCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentPDFParametersUsedCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentPDFParametersUsedCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentPDFParametersUsedCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentPDFParametersUsedCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentPDFParametersUsedCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentPDFParametersUsedCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentPDFHeatMapCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentPDFHeatMapCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentPDFHeatMapCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentPDFHeatMapCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentPDFHeatMapCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentPDFHeatMapCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentPDFClustersCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentPDFClustersCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentPDFClustersCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentPDFClustersCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentPDFClustersCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentPDFClustersCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentPDFExpRatioCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentPDFExpRatioCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentPDFExpRatioCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentPDFExpRatioCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentPDFExpRatioCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentPDFExpRatioCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentPNGHeatMapCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentPNGHeatMapCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentPNGHeatMapCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentPNGHeatMapCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
```

```
       contentPNGHeatMapCheckBox.setRolloverSelectedIcon(new
       javax.swing.ImageIcon("src/img/box_checked_hov.png"));
       contentPNGHeatMapCheckBox.setSelectedIcon(new
       javax.swing.ImageIcon("src/img/box_checked_def.png"));
}
contentXLSClustersCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentXLSClustersCheckBox.setText("Cluster Properties");
contentTXTExpRatioCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentTXTExpRatioCheckBox.setText("Expression Ratios");
contentTXTClustersCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentTXTClustersCheckBox.setText("Cluster Properties");
contentTXTClustersCheckBox.setEnabled(false);
contentTXTClustersCheckBox.setOpaque(false);
contentTXTParametersUsedCheckBox.setFont(new java.awt.Font("Tahoma",0,10));
contentTXTParametersUsedCheckBox.setText("Parameters Used");
contentTXTParametersUsedCheckBox.setEnabled(false);
contentTXTParametersUsedCheckBox.setOpaque(false);
contentXLSExpRatioCheckBox.setEnabled(false);
contentXLSExpRatioCheckBox.setOpaque(false);
contentXLSClustersCheckBox.setEnabled(false);
contentXLSClustersCheckBox.setOpaque(false);
contentTXTExpRatioCheckBox.setEnabled(false);
contentTXTExpRatioCheckBox.setOpaque(false);
contentPDFParametersUsedCheckBox.setFont(new java.awt.Font("Tahoma",0,10));
contentPDFParametersUsedCheckBox.setText("Parameters Used");
contentPDFParametersUsedCheckBox.setEnabled(false);
contentPDFParametersUsedCheckBox.setOpaque(false);
contentPDFHeatMapCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPDFHeatMapCheckBox.setText("Heatmap");
contentPDFHeatMapCheckBox.setEnabled(false);
contentPDFHeatMapCheckBox.setOpaque(false);
contentPDFClustersCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPDFClustersCheckBox.setText("Cluster Properties");
contentPDFClustersCheckBox.setEnabled(false);
contentPDFClustersCheckBox.setOpaque(false);
contentPDFExpRatioCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPDFExpRatioCheckBox.setText("Expression Ratios");
contentPDFExpRatioCheckBox.setEnabled(false);
contentPDFExpRatioCheckBox.setOpaque(false);
contentPNGHeatMapCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPNGHeatMapCheckBox.setSelected(true);
contentPNGHeatMapCheckBox.setEnabled(false);
contentPNGHeatMapCheckBox.setOpaque(false);

 exportPDFFileName.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
 exportPDFFileName.setEnabled(false);
 exportPDFFileName.setOpaque(false);
 pdfLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
 pdfLabel.setText(".pdf");
 pdfLabel.setEnabled(false);
 txtLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
 txtLabel.setText(".txt");
 txtLabel.setEnabled(false);
 exportTextFileName.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
 exportTextFileName.setEnabled(false);
 exportTextFileName.setOpaque(false);
 xlsLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
 xlsLabel.setText(".xls");
 xlsLabel.setEnabled(false);
 exportExcelFileName.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
 exportExcelFileName.setEnabled(false);
 exportExcelFileName.setOpaque(false);
 pngLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
```

```
        pngLabel.setText(".png");
        pngLabel.setEnabled(false);
        exportImageFileName.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
        exportImageFileName.setEnabled(false);
        exportImageFileName.setOpaque(false);
        contentPNGHeatmapLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
        contentPNGHeatmapLabel.setText("Heatmap");
        contentPNGHeatmapLabel.setEnabled(false);
        jLabel1.setText("Export the results into selected files.");
        status.setForeground(java.awt.Color.BLUE);
        status.setText("\t");

        javax.swing.GroupLayout form4Layout = new javax.swing.GroupLayout(form4);
        form4.setLayout(form4Layout);
        form4Layout.setHorizontalGroup(
            form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(form4Layout.createSequentialGroup().addGap(22, 22, 22)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(form4Layout.createSequentialGroup().addGap(10, 10, 10)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(form4Layout.createSequentialGroup()
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(form4Layout.createSequentialGroup().addComponent(fileFormatsLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 83,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(99, 99, 99)
.addComponent(fileNamesLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 127,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGroup(form4Layout.createSequentialGroup().
addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false).addComponent(exportToImageCheckbox, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(exportToExcelCheckbox,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(exportToTextCheckbox, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(exportToPDFCheckbox,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED_SIZE, 83,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(99, 99, 99)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(form4Layout.createSequentialGroup()
.addComponent(exportImageFileName, javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(pngLabel)).addGroup(form4Layout.createSequentialGroup()
.addComponent(exportExcelFileName, javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(xlsLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGroup(form4Layout.createSequentialGroup().
addComponent(exportTextFileName,javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(txtLabel)).addGroup(form4Layout.createSequentialGroup()
.addComponent(exportPDFFileName, javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.RELATED).addComponent(pdfLabel))))).addGap(148, 148,
148).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G).addGroup(form4Layout.createSequentialGroup().addGroup(form4Layout.createParallelGro
up(javax.swing.GroupLayout.Alignment.TRAILING,
false).addComponent(contentXLSClustersCheckBox, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(contentTXTParametersUsedCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(contentTXTClustersCheckBox)).addGap(18, 18, 18)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddComponent(contentTXTExpRatioCheckBox, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                             .addComponent(contentXLSExpRatioCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))).addGroup(form4Layout.createSequentialGroup().addGroup(form4Layout.c
reateParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
.addGroup(form4Layout.createSequentialGroup().addComponent(contentPNGHeatMapCheckBox).
addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(contentPNGHeatmapLabel)).addComponent(contentsLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 246,
javax.swing.GroupLayout.PREFERRED_SIZE).addGroup(form4Layout.createSequentialGroup().a
ddGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).add
Component(contentPDFParametersUsedCheckBox, javax.swing.GroupLayout.PREFERRED_SIZE,
104, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(contentPDFHeatMapCheckBox, javax.swing.GroupLayout.PREFERRED_SIZE, 104,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(21, 21,
21).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
).addComponent(contentPDFClustersCheckBox) .addComponent(contentPDFExpRatioCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE)))).addGap(0, 78, Short.MAX_VALUE)))
.addGap(21, 21, 21)) .addGroup(form4Layout.createSequentialGroup()
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 372,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
.addGroup(form4Layout.createSequentialGroup().addGroup(form4Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING).addComponent(errorMessageForm4,
javax.swing.GroupLayout.PREFERRED_SIZE, 483,
javax.swing.GroupLayout.PREFERRED_SIZE).addGroup(form4Layout.createSequentialGroup().a
ddComponent(destFolderLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 108,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(destFolderPath, javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.RELATED).addComponent(chooseDestFolderButton)))
.addContainerGap()).addGroup(form4Layout.createSequentialGroup()

.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddComponent(status).addComponent(progressbar, javax.swing.GroupLayout.PREFERRED_SIZE,
426, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE) .addComponent(resetButtonForm4,
javax.swing.GroupLayout.PREFERRED_SIZE, 111, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(exportButton, javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(95, 95, 95)))));

form4Layout.setVerticalGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(form4Layout.createSequentialGroup()
.addGap(28, 28, 28).addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).
addComponent(fileFormatsLabel).addComponent(fileNamesLabel).addComponent(contentsLabel
)).addGap(18, 18, 18)

.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false).addComponent(exportToImageCheckbox, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(contentPNGHeatMapCheckBox,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(exportImageFileName).addComponent(pngLabel)).addComponent(contentPNGHeat
mapLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(18, 18, 18)

.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).ad
dComponent(contentXLSClustersCheckBox).addComponent(contentXLSExpRatioCheckBox)).addGr
oup(javax.swing.GroupLayout.Alignment.TRAILING,
form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(exportToExcelCheckbox).addComponent(exportExcelFileName,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(xlsLabel)))
.addGap(18, 18, 18)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(form4Layout.createSequentialGroup().addGroup(form4Layout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.BASELINE).addComponent(exportTextFileName,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(txtLabel).addComponent(exportToTextCheckbox)).addGap(0, 0,
Short.MAX_VALUE)).addGroup(form4Layout.createSequentialGroup()
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).
addComponent(contentTXTParametersUsedCheckBox, javax.swing.GroupLayout.PREFERRED_SIZE,
21, javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(contentTXTExpRatioCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 21, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(contentTXTClustersCheckBox, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(12, 12, 12)

.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).
addComponent(contentPDFClustersCheckBox).addComponent(contentPDFParametersUsedCheckBox
, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE) .addComponent(exportPDFFileName,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(pdfLabel).addComponent(exportToPDFCheckbox))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).
addComponent(contentPDFHeatMapCheckBox, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(contentPDFExpRatioCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)))).addGroup(form4Layout.createParallelGroup(ja
vax.swing.GroupLayout.Alignment.BASELINE).addComponent(destFolderPath,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(chooseDestFolderButton).addComponent(destFolderLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(errorMessageForm4, javax.swing.GroupLayout.PREFERRED_SIZE, 17,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18)

.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING).
addGroup(form4Layout.createSequentialGroup().addComponent(status)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(progressbar, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).
addComponent(resetButtonForm4, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(exportButton)))
.addGap(36, 36, 36)));

tabbedPane.addTab("Export", form4);
layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,

       layout.createSequentialGroup().addContainerGap()
       .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
       ).addComponent(filler1, javax.swing.GroupLayout.DEFAULT_SIZE,
       javax.swing.GroupLayout.DEFAULT_SIZE,
       Short.MAX_VALUE).addComponent(tabbedPane)).addContainerGap()) );

       layout.setVerticalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alig
       nment.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
       layout.createSequentialGroup().addComponent(filler1,
       javax.swing.GroupLayout.PREFERRED_SIZE, 11,
       javax.swing.GroupLayout.PREFERRED_SIZE)

       .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
       .addComponent(tabbedPane, javax.swing.GroupLayout.PREFERRED_SIZE, 470,
       javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap(javax.swing.GroupLayout
       .DEFAULT_SIZE, Short.MAX_VALUE)));
       pack();
       this.setLocationRelativeTo(null);
}

private void exportButtonActionPerformed(java.awt.event.ActionEvent evt) {
       form4.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
       exportFile = new exportFileWorker();
       exportFile.execute();
       }

private void resetButtonForm4ActionPerformed(java.awt.event.ActionEvent evt){
       contentXLSClustersCheckBox.setSelected(false);
       contentXLSExpRatioCheckBox.setSelected(false);
       contentPDFClustersCheckBox.setSelected(false);
       contentPDFExpRatioCheckBox.setSelected(false);
       contentPDFHeatMapCheckBox.setSelected(false);
       contentPDFParametersUsedCheckBox.setSelected(false);
       contentTXTClustersCheckBox.setSelected(false);
       contentTXTExpRatioCheckBox.setSelected(false);
       contentTXTParametersUsedCheckBox.setSelected(false);
       contentXLSClustersCheckBox.setEnabled(false);
       contentXLSExpRatioCheckBox.setEnabled(false);
       contentPDFClustersCheckBox.setEnabled(false);
       contentPDFExpRatioCheckBox.setEnabled(false);
       contentPDFHeatMapCheckBox.setEnabled(false);
       contentPDFParametersUsedCheckBox.setEnabled(false);
       contentPNGHeatmapLabel.setEnabled(false);
       contentTXTClustersCheckBox.setEnabled(false);
       contentTXTExpRatioCheckBox.setEnabled(false);
       contentTXTParametersUsedCheckBox.setEnabled(false);
       exportExcelFileName.setText("");
       exportImageFileName.setText("");
       exportPDFFileName.setText("");
       exportTextFileName.setText("");
       exportExcelFileName.setEnabled(false);
```

```
        exportImageFileName.setEnabled(false);
        exportPDFFileName.setEnabled(false);
        exportTextFileName.setEnabled(false);
        pngLabel.setEnabled(false);
        txtLabel.setEnabled(false);
        pdfLabel.setEnabled(false);
        xlsLabel.setEnabled(false);
        exportToExcelCheckbox.setSelected(false);
        exportToImageCheckbox.setSelected(false);
        exportToPDFCheckbox.setSelected(false);
        exportToTextCheckbox.setSelected(false);
        destFolderPath.setText("");
        errorMessageForm4.setText("");
        status.setText("\t");
        progressbar.setValue(0);
        if(exportFile != null)exportFile.cancel(true);
}

private void chooseDestFolderButtonActionPerformed(java.awt.event.ActionEvent evt) {
        folderChooser.setCurrentDirectory(new java.io.File("."));
        folderChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        folderChooser.setAcceptAllFileFilterUsed(false);
        if (folderChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
                destFolderPath.setText(""+folderChooser.getSelectedFile());
}

private void exportToImageCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
        exportImageFileName.setEnabled(exportToImageCheckbox.isSelected());
        contentPNGHeatmapLabel.setEnabled(exportToImageCheckbox.isSelected());
        pngLabel.setEnabled(exportToImageCheckbox.isSelected());
}

private void exportToPDFCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
        contentPDFClustersCheckBox.setEnabled(exportToPDFCheckbox.isSelected());
        contentPDFExpRatioCheckBox.setEnabled(exportToPDFCheckbox.isSelected());
        contentPDFHeatMapCheckBox.setEnabled(exportToPDFCheckbox.isSelected());
        contentPDFParametersUsedCheckBox.setEnabled(exportToPDFCheckbox.isSelected());
        exportPDFFileName.setEnabled(exportToPDFCheckbox.isSelected());
        pdfLabel.setEnabled(exportToPDFCheckbox.isSelected());
}

private void exportToExcelCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
        contentXLSClustersCheckBox.setEnabled(exportToExcelCheckbox.isSelected());
        contentXLSExpRatioCheckBox.setEnabled(exportToExcelCheckbox.isSelected());
        exportExcelFileName.setEnabled(exportToExcelCheckbox.isSelected());
        xlsLabel.setEnabled(exportToExcelCheckbox.isSelected());
}

private void exportToTextCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
        contentTXTClustersCheckBox.setEnabled(exportToTextCheckbox.isSelected());
        contentTXTExpRatioCheckBox.setEnabled(exportToTextCheckbox.isSelected());
        contentTXTParametersUsedCheckBox.setEnabled(exportToTextCheckbox.isSelected());
        exportTextFileName.setEnabled(exportToTextCheckbox.isSelected());
        txtLabel.setEnabled(exportToTextCheckbox.isSelected());
}

class exportFileWorker extends SwingWorker<String, Void> {
        @Override public String doInBackground() {
                FileExporter fileExporter = new FileExporter();
                if(!(exportToImageCheckbox.isSelected() ||
                exportToExcelCheckbox.isSelected() || exportToPDFCheckbox.isSelected() ||
                exportToTextCheckbox.isSelected()))
                        return "Please choose at least one file to export.";
                else if(destFolderPath.getText().equals("")) return "Select where
                to put the files.";
                else if (!fileExporter.canWriteToDirectory(destFolderPath
                .getText()+"\\temp.txt")) return "Invalid destination folder.";
                else{
                        if(exportToImageCheckbox.isSelected()){ //image
                                if(exportImageFileName.getText().equals(""))
                                        return "File name for the image is required.";
                                else
                                if(!fileExporter.isFilenameValid(exportImageFileName.getTex
                                t()+".png"))
                                        return "File name for the image is invalid.";
                                else{ // generate image
                                        if(this.isCancelled()) return "";
                                        status.setText("Exporting image file...");
                                        progressbar.setValue(5);
                                        if(!fileExporter.saveHeatmapPanelToImage(heatmap,dest
                                        FolderPath.getText()+"\\"+exportImageFileName.getText
                                        ()+".png"))
                                                return "Problem occured while exporting the
                                                image";
                                        progressbar.setValue(25);
                                        }
                                }
                                if(exportToExcelCheckbox.isSelected()){//excel
                                        if(exportExcelFileName.getText().equals(""))
                                                return "File name for the spreadsheet is
                                                required.";
                                        else if(!fileExporter.isFilenameValid
                                        (exportExcelFileName.getText()+".xls"))
                                                return "File name for the spreadsheet is
                                                invalid.";
                                        else if(contentXLSClustersCheckBox.isSelected() ||
                                        contentXLSExpRatioCheckBox.isSelected()){
                                                if(this.isCancelled()) return "";
                                                status.setText("Exporting spreadsheet...");
                                                progressbar.setValue(30);
                                                ArrayList<JTable> tables = new ArrayList<>();
                                                // generate excel: clusters or exp ratio
if(contentXLSClustersCheckBox.isSelected()){
        JTable temp = new JTable(new DefaultTableModel(new Object[][]{
        {"Number of clusters", numOfClustStr},
        {"Mean Squared Distances of Centroids", overAllVariance} },
        new Object[]{"",""} ));
        tables.add(temp);
        tables.add(clustPropertiesTableWithOutButtons);
        if(contentXLSExpRatioCheckBox.isSelected()) tables.add(expRatioTable);
        if(!fileExporter.exportTablesToFile(tables,
        destFolderPath.getText()+"\\"+exportExcelFileName.getText()+".xls"))
                return "Problem occured while exporting the spreadsheet.";
        progressbar.setValue(50);
        }else return "Choose what contents to include in the spreadsheet.";
}
if(exportToTextCheckbox.isSelected()){//text
        if(exportTextFileName.getText().equals("")) return "File name for the text file
        is required.";
        else if(!fileExporter.isFilenameValid(exportTextFileName .getText()+".txt"))
        return "File name for the text file is invalid.";
        else if( contentTXTParametersUsedCheckBox.isSelected() ||
        contentTXTExpRatioCheckBox.isSelected() ||
        contentTXTClustersCheckBox.isSelected()){
                if(this.isCancelled()) return "";
                status.setText("Exporting text file...");
```

```
            progressbar.setValue(55);
            ArrayList<JTable> tables = new ArrayList<>();// generate text file:
            params or clusters or exp ratio
        if(contentTXTParametersUsedCheckBox.isSelected()) tables.add(paramUsedTable);
        if(contentTXTClustersCheckBox.isSelected()){
            JTable temp = new JTable(new DefaultTableModel(new Object[][]{
            {"Number of clusters", numOfClustStr},
            {"Mean Squared Distances of Centroids", overAllVariance} },
            new Object[]{"",""} ));
            tables.add(temp);
            tables.add(clustPropertiesTableWithOutButtons);
        }
        if(contentTXTExpRatioCheckBox.isSelected()) tables.add(expRatioTable);
        if(!fileExporter.exportTablesToFile(tables,
        destFolderPath.getText()+"\\"+exportTextFileName.getText()+".txt"))
            return "Problem occured while exporting the text file.";
                progressbar.setValue(75);
        }else return "Choose what contents to include in the text file.";
    }
    if(exportToPDFCheckbox.isSelected()){//pdf
        if(exportPDFFileName.getText().equals("")) return "File name for the PDF file
        is required.";
        else if(!fileExporter.isFilenameValid(exportPDFFileName .getText()+".pdf"))
        return "File name for the PDF file is invalid.";
        else if( contentPDFParametersUsedCheckBox.isSelected() ||
        contentPDFExpRatioCheckBox.isSelected() ||
        contentPDFClustersCheckBox.isSelected() ||
        contentPDFHeatMapCheckBox.isSelected() ){
            if(this.isCancelled()) return "";
            status.setText("Exporting PDF file...");
            progressbar.setValue(80);
                        ArrayList<JTable> tables = new ArrayList<>();//
                        generate PDF file: params or heatmap or clusters or
                        exp ratio
                        if(contentPDFParametersUsedCheckBox.isSelected())
                        tables.add(paramUsedTable);
                        if(contentPDFClustersCheckBox.isSelected()){
                        JTable temp = new JTable(new DefaultTableModel(new
                        Object[][]{   {"Number of clusters", numOfClustStr},
                            Mean Squared Distances of Centroids",
                            overAllVariance} },
                            new Object[]{"Results",""} ));
                            tables.add(temp);
                            tables.add(clustPropertiesTableWithOutButtons)
                            ;
                        }
                        if(contentPDFExpRatioCheckBox.isSelected())
                            tables.add(expRatioTable);
                        JPanel temp = null;
                        if(contentPDFHeatMapCheckBox.isSelected())
                            temp = heatmap;
                        if(!fileExporter.exportToPDF(tables,
                        destFolderPath.getText()+
                        "\\"+exportPDFFileName.getText()+".pdf", temp))
                            return "Problem occured while exporting
                             the PDF file.";
                    }else return "Choose what contents to include
                         in the PDF file.";
                }
            }
            return "";
        }
        @Override public void done() {
            Toolkit.getDefaultToolkit().beep();
            try {
                  errorMessageForm4.setText(get());
            } catch(CancellationException e){
                  status.setText("File exporting is cancelled.");
                  progressbar.setValue(0);
            } catch (InterruptedException | ExecutionException e) {
                  errorMessageForm4.setText("Problem occured.");
            }
            if(errorMessageForm4.getText().equals("")
             && !this.isCancelled()){
                  status.setText("Done exporting.");
                  progressbar.setValue(100);
            }
            form4.setCursor(Cursor.getPredefinedCursor
            (Cursor.DEFAULT_CURSOR));
            }
      }
}
```

K. MaSOM/src/JTableModel.java

```
import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.net.URL;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.TableCellRenderer;

public class JTableModel extends AbstractTableModel {
      private static final long serialVersionUID = 1L;
      private String[] columnNames;
      private Class<?>[] columnTypes;
      private Object[][] contents;
      private String[][] description;
      private String[] geneIDs, sampleIDs;
      private int[] clusterAssignments;
      private double[][] data;
      private FileExporter fe = new FileExporter();
      private JFileChooser fileChooser = new JFileChooser();
      private boolean clusterSamples;

      public JTableModel(String[] cols, Object[][] contents){
            columnNames = new String[cols.length +2];
            columnNames[0] = "Data";
            columnNames[1] = "Description";
            for(int i=0; i<cols.length; i++){
                  columnNames[i+2] = cols[i];
            }
            columnTypes = new Class[cols.length +2];
            columnTypes[0] =  JButton.class;
```

```
        columnTypes[1] =  JButton.class;
        for(int i=0; i<cols.length; i++){
            columnTypes[i+2] = Object.class;
        }
        this.contents = contents;
    }

    public void setExportData(double[][] data,int[] clusterAssignment,boolean
    clusterSamples,String[] geneIDs,String[] sampleIDs, String[][] description){
        this.data = data;
        this.clusterAssignments = clusterAssignment;
        this.clusterSamples =clusterSamples;
        this.geneIDs = geneIDs;
        this.sampleIDs = sampleIDs;
        this.description = description;
    }

    public boolean isCellEditable(int rowIndex, int columnIndex){
        return (columnIndex == 5) ;

    }
    @Override public int getColumnCount() {
        return columnNames.length;
    }

    @Override public int getRowCount() {
        return contents.length;
    }

    @Override public String getColumnName(int columnIndex) {
        return columnNames[columnIndex];
    }

    @Override public Class<?> getColumnClass(int columnIndex) {
        return columnTypes[columnIndex];
    }

    @Override public Object getValueAt(final int rowIndex, final int columnIndex) {
        if (columnIndex == 0){
            final JButton button;
            final ImageIcon icon;
            URL imgUrl = getClass().getResource ("src/img/download_def.png");
            if(imgUrl !=null){
                button = new JButton(new ImageIcon(imgUrl));
                imgUrl = getClass().getResource("src/img/main.png");
                icon = new ImageIcon(imgUrl);
            }
            else{
                button = new JButton(new ImageIcon
                ("src/img/download_def.png"));
                icon = new ImageIcon("src/img/main.png");
            }
            button.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) {
                    int returnVal = fileChooser.showSaveDialog(
                        new JFrame());
                    if (returnVal == JFileChooser.APPROVE_OPTION) {
                        String f = fileChooser.getSelectedFile()
                        .getAbsolutePath();
                        if(!f.endsWith(".txt")) f += ".txt";
    if(fe.exportClusterDataToFile(data,clusterAssignments,
    clusterSamples, geneIDs, sampleIDs, rowIndex, f))
```

```
              JOptionPane.showMessageDialog(null, "Cluster "+(rowIndex+1)+ " data was
              saved.", "Success", JOptionPane.PLAIN_MESSAGE, icon);
       else JOptionPane.showMessageDialog(null, "Cluster "+(rowIndex+1)+ "
               data was not saved.", "Error", JOptionPane.PLAIN_MESSAGE, icon);
                                   }
                            }
                     });
                     return button;
              }
              if (columnIndex == 1){
                     final JButton button;
                     final ImageIcon icon;
                     URL imgUrl = getClass().getResource( "src/img/download_def.png");
                     if(imgUrl !=null){
                            button = new JButton(new ImageIcon(imgUrl));
                            imgUrl = getClass().getResource(
                            "src/img/download_hov.png");
                            button.setRolloverIcon(new ImageIcon(imgUrl));
                            imgUrl = getClass().getResource("src/img/main.png");
                            icon = new ImageIcon(imgUrl);
                     }else{
                            button = new JButton(new ImageIcon(
                            "src/img/download_def.png"));
                            button.setRolloverIcon(new ImageIcon(
                            "src/img/download_hov.png"));
                            icon = new ImageIcon("src/img/main.png");
                     }
                     button.addActionListener(new ActionListener() {
                            public void actionPerformed(ActionEvent arg0) {
                                   int returnVal = fileChooser.showSaveDialog(
                                           new JFrame());
                                   if (returnVal == JFileChooser.APPROVE_OPTION) {
                                          String f = fileChooser.getSelectedFile()
                                          .getAbsolutePath();
                                          if(!f.endsWith(".txt")) f += ".txt";

       if(fe.exportClusterDescToFile(description,clusterAssignments,
       rowIndex, f))
              JOptionPane.showMessageDialog(null, "Cluster "+(rowIndex+1)+ "
              description was saved.", "Success",
              JOptionPane.PLAIN_MESSAGE, icon);
       else JOptionPane.showMessageDialog(null, "Cluster "+(rowIndex+1)+ "
               description was not saved.", "Error",
               JOptionPane.PLAIN_MESSAGE, icon);
                                   }
                            }
                     });
                     return button;
              }
              else return contents[rowIndex][columnIndex-2];
       }
}

class JTableButtonRenderer implements TableCellRenderer {
@Override public Component getTableCellRendererComponent(JTable table, Object value,
boolean isSelected, boolean hasFocus, int row, int column) {
       JButton button = (JButton)value;
       return button;
       }
}

class JTableButtonMouseListener extends MouseAdapter {
       private final JTable table;
```

```
	public JTableButtonMouseListener(JTable table) {
		this.table = table;
	}
	public void mouseClicked(MouseEvent e) {
		int column = table.getColumnModel().getColumnIndexAtX(e.getX());
		int row = e.getY()/table.getRowHeight();
		if (row < table.getRowCount() && row >= 0 && column <
table.getColumnCount() && column >= 0) {
			Object value = table.getValueAt(row, column);
			if (value instanceof JButton) ((JButton)value).doClick();
		}
	}
}
```

L. MaSOM/src/UserInterface.java

```
import java.awt.Color;
import java.awt.Cursor;
import java.awt.Toolkit;
import java.io.File;
import java.net.URL;
import java.util.ArrayList;
import java.util.concurrent.CancellationException;
import java.util.concurrent.ExecutionException;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.SwingWorker;
import javax.swing.table.DefaultTableModel;

@SuppressWarnings("serial")
public class UserInterface extends javax.swing.JFrame {
	String formTitles[] = {	"Submit ", "Preprocess ", "Cluster ", "Export "
	};
	String formDescriptions[] = {
	"<html><p align=right>Import files containing the expression ratios.
Supplementary data for genes and samples may optionally be supplied for the
documentation of results. All files must follow the format specified in the
guidelines.</html>", "<html><p align=right>Remove experimental bias and non-biological
variations by optionally preprocessing the expression ratios.</html>","<html><p
align=right>Group genes or samples using Self-Organizing Maps. Feel free to modify the
parameters to be used for clustering.</html>","<html><p align=right>Explore the
clustered data through the provided summary and visualization. Results may be exported
into available file formats for further analysis and archiving.</html>"
};
	String formImages[] = { "src/img/big_1.png", "src/img/big_2.png",
"src/img/big_3.png","src/img/big_4.png"};

public UserInterface() {
		URL imgUrl = getClass().getResource("src/img/main.png");
	if(imgUrl !=null) this.setIconImage(new ImageIcon(imgUrl).getImage());
	else this.setIconImage(new ImageIcon("src/img/main.png").getImage());
		initComponents();

		genesOrSamplesgroup.add(clusterByGenesRadio);
		genesOrSamplesgroup.add(clusterBySamplesRadio);
		distMetricGroup.add(manhattanRadio);
		distMetricGroup.add(euclidianRadio);
```

```
        distMetricGroup.add(supnormRadio);
        form2.setVisible(false);
        form3.setVisible(false);
        form4.setVisible(false);
        viewHeatMapButton.setVisible(false);
        viewDataButton.setVisible(false);
        nextButton.setEnabled(false);
        errorMessageForm1.setVisible(false);
        errorMessageForm2.setVisible(false);
        errorMessageForm3.setVisible(false);
        errorMessageForm4.setVisible(true);
        status.setForeground(Color.BLUE);
        status.setText("\t");
        status.setVisible(true);
        helpWindow.setVisible(false);
}

private void initComponents() {
        genesOrSamplesgroup = new javax.swing.ButtonGroup();
        distMetricGroup = new javax.swing.ButtonGroup();
        backgroundPanel = new javax.swing.JPanel();
        Banner = new javax.swing.JLabel();
        DataIcon = new javax.swing.JLabel();
        PreprocessIcon = new javax.swing.JLabel();
        ClusterIcon = new javax.swing.JLabel();
        VisualizeIcon = new javax.swing.JLabel();
        mainContentPanel = new javax.swing.JPanel();
        jPanel3 = new javax.swing.JPanel();
        jLayeredPane1 = new javax.swing.JLayeredPane();
        stepDescription = new javax.swing.JLabel();
        stepTitle = new javax.swing.JLabel();
        viewDataButton = new javax.swing.JButton();
        viewHeatMapButton = new javax.swing.JButton();
        stepBigImageLabel = new javax.swing.JLabel();
        progressPanel = new javax.swing.JPanel();
        progressbar = new javax.swing.JProgressBar();
        status = new javax.swing.JLabel();
        navigationButtonsPanel = new javax.swing.JPanel();
        backButton = new javax.swing.JButton();
        nextButton = new javax.swing.JButton();
        contentPane = new javax.swing.JLayeredPane();
        form4 = new javax.swing.JPanel();
        exportButton = new javax.swing.JButton();
        errorMessageForm4 = new javax.swing.JLabel();
        resetButtonForm4 = new javax.swing.JButton();
        chooseDestFolderButton = new javax.swing.JButton();
        destFolderLabel = new javax.swing.JLabel();
        destFolderPath = new javax.swing.JTextField();
        exportToImageCheckbox = new javax.swing.JCheckBox();
        exportToPDFCheckbox = new javax.swing.JCheckBox();
        exportToExcelCheckbox = new javax.swing.JCheckBox();
        exportToTextCheckbox = new javax.swing.JCheckBox();
        fileFormatsLabel = new javax.swing.JLabel();
        contentsLabel = new javax.swing.JLabel();
        fileNamesLabel = new javax.swing.JLabel();
        contentXLSExpRatioCheckBox = new javax.swing.JCheckBox();
        contentXLSClustersCheckBox = new javax.swing.JCheckBox();
        contentTXTExpRatioCheckBox = new javax.swing.JCheckBox();
        contentTXTClustersCheckBox = new javax.swing.JCheckBox();
        contentTXTParametersUsedCheckBox = new javax.swing.JCheckBox();
        contentPDFParametersUsedCheckBox = new javax.swing.JCheckBox();
        contentPDFHeatMapCheckBox = new javax.swing.JCheckBox();
        contentPDFClustersCheckBox = new javax.swing.JCheckBox();
```

```
contentPDFExpRatioCheckBox = new javax.swing.JCheckBox();
contentPNGHeatMapCheckBox = new javax.swing.JCheckBox();
exportPDFFileName = new javax.swing.JTextField();
pdfLabel = new javax.swing.JLabel();
txtLabel = new javax.swing.JLabel();
exportTextFileName = new javax.swing.JTextField();
xlsLabel = new javax.swing.JLabel();
exportExcelFileName = new javax.swing.JTextField();
pngLabel = new javax.swing.JLabel();
exportImageFileName = new javax.swing.JTextField();
contentPNGHeatmapLabel = new javax.swing.JLabel();
form3 = new javax.swing.JPanel();
form3Instruction = new javax.swing.JLabel();
clusterButton = new javax.swing.JButton();
errorMessageForm3 = new javax.swing.JLabel();
resetButtonForm3 = new javax.swing.JButton();
numberOfClustersLabel = new javax.swing.JLabel();
clusterByGenesRadio = new javax.swing.JRadioButton();
clusterBySamplesRadio = new javax.swing.JRadioButton();
numOfClustText = new javax.swing.JTextField();
distMetricLabel = new javax.swing.JLabel();
manhattanRadio = new javax.swing.JRadioButton();
euclidianRadio = new javax.swing.JRadioButton();
supnormRadio = new javax.swing.JRadioButton();
initLearningRateLabel = new javax.swing.JLabel();
initLearningRateText = new javax.swing.JTextField();
maxIterLabel = new javax.swing.JLabel();
maxIterText = new javax.swing.JTextField();
footnoteLabel2 = new javax.swing.JLabel();
form2 = new javax.swing.JPanel();
form2Instruction = new javax.swing.JLabel();
preprocessButton = new javax.swing.JButton();
errorMessageForm2 = new javax.swing.JLabel();
resetButtonForm2 = new javax.swing.JButton();
logTransCheckbox = new javax.swing.JCheckBox();
zScoreTransCheckbox = new javax.swing.JCheckBox();
medianCenteringCheckbox = new javax.swing.JCheckBox();
scaleNormCheckbox = new javax.swing.JCheckBox();
quantileNormCheckbox = new javax.swing.JCheckBox();
logTransHelp = new javax.swing.JButton();
medianCenteringHelp = new javax.swing.JButton();
zScoreTransHelp = new javax.swing.JButton();
scaleNormHelp = new javax.swing.JButton();
quantileNormHelp = new javax.swing.JButton();
form1 = new javax.swing.JPanel();
form1Instruction = new javax.swing.JLabel();
dataFileLabel = new javax.swing.JLabel();
sampleDescriptionLabel = new javax.swing.JLabel();
geneDescriptionLabel = new javax.swing.JLabel();
dataFilePath = new javax.swing.JTextField();
geneFilePath = new javax.swing.JTextField();
sampleFilePath = new javax.swing.JTextField();
chooseDataFileButton = new javax.swing.JButton();
chooseGeneFileButton = new javax.swing.JButton();
chooseSampleFileButton = new javax.swing.JButton();
dataFileHelp = new javax.swing.JButton();
geneFileHelp = new javax.swing.JButton();
sampleFileHelp = new javax.swing.JButton();
uploadFileButton = new javax.swing.JButton();
footnoteLabel1 = new javax.swing.JLabel();
errorMessageForm1 = new javax.swing.JLabel();
resetButtonForm1 = new javax.swing.JButton();
mainHelpButton = new javax.swing.JButton();
```

```
                setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("MaSOM: Clustering Microarray Data Using
        Self-Organizing Maps");
                setBackground(new java.awt.Color(255, 255, 255));
                setPreferredSize(new java.awt.Dimension(846, 590));
                setResizable(false);
                backgroundPanel.setBackground(new java.awt.Color(255, 255, 255));
                URL imgUrl = getClass().getResource("src/img/banner.png");
                if(imgUrl !=null) Banner.setIcon(new ImageIcon(imgUrl));
                else Banner.setIcon(new javax.swing.ImageIcon("src/img/banner.png"));
                DataIcon.setBackground(new java.awt.Color(255, 255, 255));
                imgUrl = getClass().getResource("src/img/step1_act.png");
                if(imgUrl !=null) DataIcon.setIcon(new ImageIcon(imgUrl));
                else DataIcon.setIcon(new
                javax.swing.ImageIcon("src/img/step1_act.png"));
                DataIcon.setOpaque(true);
                PreprocessIcon.setBackground(new java.awt.Color(255, 255, 255));
                imgUrl = getClass().getResource("src/img/step2_act.png");
                if(imgUrl !=null) PreprocessIcon.setIcon(new ImageIcon(imgUrl));
                else PreprocessIcon.setIcon(new
                javax.swing.ImageIcon("src/img/step2_act.png"));
                PreprocessIcon.setEnabled(false);
                PreprocessIcon.setOpaque(true);
                ClusterIcon.setBackground(new java.awt.Color(255, 255, 255));
                imgUrl = getClass().getResource("src/img/step3_act.png");
                if(imgUrl !=null) ClusterIcon.setIcon(new ImageIcon(imgUrl));
                else ClusterIcon.setIcon(new
                javax.swing.ImageIcon("src/img/step3_act.png"));
                ClusterIcon.setEnabled(false);
                ClusterIcon.setOpaque(true);
                VisualizeIcon.setBackground(new java.awt.Color(255, 255, 255));
                imgUrl = getClass().getResource("src/img/step4_act.png");
                if(imgUrl !=null) VisualizeIcon.setIcon(new ImageIcon(imgUrl));
                else VisualizeIcon.setIcon(new
                javax.swing.ImageIcon("src/img/step4_act.png"));
                VisualizeIcon.setEnabled(false);
                VisualizeIcon.setOpaque(true);
                mainContentPanel.setOpaque(false);
                jLayeredPane1.setBackground(new java.awt.Color(255, 255, 255));
                jLayeredPane1.setOpaque(true);
                stepDescription.setFont(new java.awt.Font("Tahoma", 0, 10));
                stepDescription.setForeground(new java.awt.Color(50, 59, 50));
                stepDescription.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        stepDescription.setText("<html><p align=right>Import files containing the
expression ratios. Supplementary data for genes and samples may optionally be supplied
for the documentation of results. All files must follow the format specified in the
guidelines.</html>");
                stepDescription.setVerticalAlignment(javax.swing.SwingConstants.TOP);
        stepDescription.setHorizontalTextPosition (javax.swing.SwingConstants.LEADING);
        stepDescription.setVerticalTextPosition (javax.swing.SwingConstants.TOP);
        stepDescription.setBounds(40, 60, 220, 80);
        jLayeredPane1.add(stepDescription, javax.swing.JLayeredPane.DEFAULT_LAYER);
        stepTitle.setFont(new java.awt.Font("Arial Unicode MS", 2, 36));
        stepTitle.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        stepTitle.setText("Submit ");
        stepTitle.setVerticalAlignment(javax.swing.SwingConstants.BOTTOM);
        stepTitle.setAlignmentY(0.0F);
        stepTitle.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        stepTitle.setBounds(0, 10, 260, 54);
        jLayeredPane1.add(stepTitle, javax.swing.JLayeredPane.DEFAULT_LAYER);
        imgUrl = getClass().getResource("src/img/results_def.png");
        if(imgUrl !=null) viewDataButton.setIcon(new ImageIcon(imgUrl));
```

```
else viewDataButton.setIcon(new
javax.swing.ImageIcon("src/img/results_def.png"));
viewDataButton.setText("View Data");
viewDataButton.setBorder(new javax.swing.border.LineBorder(new
java.awt.Color(0, 0, 0), 1, true));
viewDataButton.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/results_hov.png");
if(imgUrl !=null) viewDataButton.setRolloverIcon(new ImageIcon(imgUrl));
else viewDataButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/results_hov.png"));
viewDataButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       viewDataButtonActionPerformed(evt);
}});
viewDataButton.setBounds(40, 330, 110, 30);
jLayeredPane1.add(viewDataButton, javax.swing.JLayeredPane.DEFAULT_LAYER);
imgUrl = getClass().getResource("src/img/heatmap_def.png");
if(imgUrl !=null) viewHeatMapButton.setIcon(new ImageIcon(imgUrl));
else viewHeatMapButton.setIcon(new
javax.swing.ImageIcon("src/img/heatmap_def.png"));
viewHeatMapButton.setText("View Heatmap");
viewHeatMapButton.setBorder(new javax.swing.border.LineBorder(new
java.awt.Color(0, 0, 0), 1, true));
viewHeatMapButton.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/heatmap_hov.png");
if(imgUrl !=null) viewHeatMapButton.setRolloverIcon(new ImageIcon(imgUrl));
else viewHeatMapButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/heatmap_hov.png"));
viewHeatMapButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       viewHeatMapButtonActionPerformed(evt);
}});
viewHeatMapButton.setBounds(160, 330, 110, 30);
jLayeredPane1.add(viewHeatMapButton, javax.swing.JLayeredPane.DEFAULT_LAYER);
stepBigImageLabel.setHorizontalAlignment
(javax.swing.SwingConstants.RIGHT);
imgUrl = getClass().getResource(formImages[0]);
if(imgUrl !=null) stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
else stepBigImageLabel.setIcon(new javax.swing.ImageIcon(formImages[0]));
       stepBigImageLabel.setVerticalAlignment(javax.swing.SwingConstants.TOP);
stepBigImageLabel.setBounds(110, 20, 160, 160);
jLayeredPane1.add(stepBigImageLabel, javax.swing.JLayeredPane.DEFAULT_LAYER);
javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(
jPanel3Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING).addComponent(jLayeredPane1,
javax.swing.GroupLayout.Alignment.TRAILING));
jPanel3Layout.setVerticalGroup(jPanel3Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel3Layout.createSequentialGroup()
.addComponent(jLayeredPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 369,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE)));
progressPanel.setOpaque(false);
javax.swing.GroupLayout progressPanelLayout = new
javax.swing.GroupLayout(progressPanel);
progressPanel.setLayout(progressPanelLayout);
progressPanelLayout.setHorizontalGroup(progressPanelLayout.
createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(progressPanelLayout.createSequentialGroup().addContainerGap()
.addGroup(progressPanelLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING).addComponent(progressbar, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
progressPanelLayout.createSequentialGroup().addGap(0, 8,
Short.MAX_VALUE).addComponent(status, javax.swing.GroupLayout.PREFERRED_SIZE,
478, javax.swing.GroupLayout.PREFERRED_SIZE)))

.addContainerGap()));
progressPanelLayout.setVerticalGroup(
progressPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
progressPanelLayout.createSequentialGroup().addContainerGap(javax.swing.GroupLa
yout.DEFAULT_SIZE, Short.MAX_VALUE).addComponent(status)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(progressbar, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(19, 19, 19)));
navigationButtonsPanel.setOpaque(false);
imgUrl = getClass().getResource("src/img/prev_def.png");
if(imgUrl !=null) backButton.setIcon(new ImageIcon(imgUrl));
else backButton.setIcon(new javax.swing.ImageIcon("src/img/prev_def.png"));
backButton.setText("Back");
backButton.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/prev_dis.png");
if(imgUrl !=null) backButton.setDisabledIcon(new ImageIcon(imgUrl));
else backButton.setDisabledIcon(new
javax.swing.ImageIcon("src/img/prev_dis.png"));
backButton.setEnabled(false);
backButton.setIconTextGap(8);
imgUrl = getClass().getResource("src/img/prev_hov.png");
if(imgUrl !=null) backButton.setRolloverIcon(new ImageIcon(imgUrl));
else backButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/prev_hov.png"));
backButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
              backButtonActionPerformed(evt);
}});
imgUrl = getClass().getResource("src/img/next_def.png");
if(imgUrl !=null) nextButton.setIcon(new ImageIcon(imgUrl));
else nextButton.setIcon(new javax.swing.ImageIcon("src/img/next_def.png"));
nextButton.setText("Next");
nextButton.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/next_dis.png");
if(imgUrl !=null) nextButton.setDisabledIcon(new ImageIcon(imgUrl));
else nextButton.setDisabledIcon(new
javax.swing.ImageIcon("src/img/next_dis.png"));
nextButton.setHorizontalTextPosition
(javax.swing.SwingConstants.LEADING);
nextButton.setIconTextGap(8);
imgUrl = getClass().getResource("src/img/next_hov.png");
if(imgUrl !=null) nextButton.setRolloverIcon(new ImageIcon(imgUrl));
else nextButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/next_hov.png"));
nextButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       nextButtonActionPerformed(evt);
}});
javax.swing.GroupLayout navigationButtonsPanelLayout = new
javax.swing.GroupLayout(navigationButtonsPanel);
navigationButtonsPanel.setLayout(navigationButtonsPanelLayout);
navigationButtonsPanelLayout.setHorizontalGroup(navigationButtonsPanelLayout.cr
eateParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
.addGroup(navigationButtonsPanelLayout.createSequentialGroup()
       .addContainerGap(83, Short.MAX_VALUE).addComponent(backButton)
       .addGap(2, 2, 2).addComponent(nextButton).addGap(4, 4, 4)));
navigationButtonsPanelLayout.setVerticalGroup(
navigationButtonsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
navigationButtonsPanelLayout.createSequentialGroup().addGap(0, 22,
Short.MAX_VALUE).addGroup(navigationButtonsPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING).addComponent(backButton,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(nextButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE))));
contentPane.setBackground(new java.awt.Color(255, 255, 255));
form4.setBackground(new java.awt.Color(255, 255, 255));
exportButton.setText("Export");
exportButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       exportButtonActionPerformed(evt);
}});
errorMessageForm4.setForeground(new java.awt.Color(255, 0, 0));
imgUrl = getClass().getResource("src/img/reset_def.png");
if(imgUrl !=null) resetButtonForm4.setIcon(new ImageIcon(imgUrl));
else resetButtonForm4.setIcon(new
javax.swing.ImageIcon("src/img/reset_def.png"));
resetButtonForm4.setText("Reset");
imgUrl = getClass().getResource("src/img/reset_hov.png");
if(imgUrl !=null) resetButtonForm4.setRolloverIcon(new ImageIcon(imgUrl));
else resetButtonForm4.setRolloverIcon(new
javax.swing.ImageIcon("src/img/reset_hov.png"));
resetButtonForm4.addActionListener
(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       resetButtonForm4ActionPerformed(evt);
}});
imgUrl = getClass().getResource("src/img/folder_def.png");
if(imgUrl !=null) chooseDestFolderButton.setIcon(new ImageIcon(imgUrl));
else chooseDestFolderButton.setIcon(new
javax.swing.ImageIcon("src/img/folder_def.png"));
chooseDestFolderButton.setText("Choose");
chooseDestFolderButton.setOpaque(false);
imgUrl = getClass().getResource("src/img/folder_hov.png");
if(imgUrl !=null) chooseDestFolderButton.setRolloverIcon(new
ImageIcon(imgUrl));
else chooseDestFolderButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/folder_hov.png"));
chooseDestFolderButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       chooseDestFolderButtonActionPerformed(evt);
}});
destFolderLabel.setText("Destination folder");
destFolderPath.setEditable(false);
destFolderPath.setOpaque(false);
exportToImageCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToImageCheckbox.setText("Image");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToImageCheckbox.setIcon(new ImageIcon(imgUrl));
else exportToImageCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToImageCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
```

```
if(imgUrl !=null) exportToImageCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToImageCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToImageCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToImageCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToImageCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToImageCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToImageCheckbox.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       exportToImageCheckboxActionPerformed(evt);
}});
exportToPDFCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToPDFCheckbox.setText("PDF");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToPDFCheckbox.setIcon(new ImageIcon(imgUrl));
       else exportToPDFCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToPDFCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
if(imgUrl !=null) exportToPDFCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToPDFCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToPDFCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToPDFCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToPDFCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToPDFCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToPDFCheckbox.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       exportToPDFCheckboxActionPerformed(evt);
}});
exportToExcelCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToExcelCheckbox.setText("Spreadsheet");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToExcelCheckbox.setIcon(new ImageIcon(imgUrl));
else exportToExcelCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToExcelCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
if(imgUrl !=null) exportToExcelCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToExcelCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToExcelCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToExcelCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
       imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToExcelCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToExcelCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToExcelCheckbox.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       exportToExcelCheckboxActionPerformed(evt);
}});
exportToTextCheckbox.setFont(new java.awt.Font("Tahoma", 0, 10));
exportToTextCheckbox.setText("Text");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null) exportToTextCheckbox.setIcon(new ImageIcon(imgUrl));
else exportToTextCheckbox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
exportToTextCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
if(imgUrl !=null) exportToTextCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
else exportToTextCheckbox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
if(imgUrl !=null) exportToTextCheckbox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
else exportToTextCheckbox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
if(imgUrl !=null) exportToTextCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
else exportToTextCheckbox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
exportToTextCheckbox.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
        exportToTextCheckboxActionPerformed(evt);
}});
fileFormatsLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
fileFormatsLabel.setText("File Format");
contentsLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
contentsLabel.setText("Contents");
fileNamesLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
fileNamesLabel.setText("File Name");
contentXLSExpRatioCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentXLSExpRatioCheckBox.setText("Expression Ratios");
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
if(imgUrl !=null){
        contentXLSExpRatioCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_dis.png");
        contentXLSExpRatioCheckBox.setDisabledSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
        contentXLSExpRatioCheckBox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        contentXLSExpRatioCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        contentXLSExpRatioCheckBox.setRolloverSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        contentXLSExpRatioCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
        contentXLSClustersCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_dis.png");
        contentXLSClustersCheckBox.setDisabledSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
        contentXLSClustersCheckBox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        contentXLSClustersCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        contentXLSClustersCheckBox.setRolloverSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        contentXLSClustersCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
```

```
contentTXTExpRatioCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentTXTExpRatioCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentTXTExpRatioCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentTXTExpRatioCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentTXTExpRatioCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentTXTExpRatioCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentTXTClustersCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentTXTClustersCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentTXTClustersCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentTXTClustersCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentTXTClustersCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentTXTClustersCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentTXTParametersUsedCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentTXTParametersUsedCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentTXTParametersUsedCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentTXTParametersUsedCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentTXTParametersUsedCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentTXTParametersUsedCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentPDFParametersUsedCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentPDFParametersUsedCheckBox.setDisabledSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentPDFParametersUsedCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentPDFParametersUsedCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_hov.png");
contentPDFParametersUsedCheckBox.setRolloverSelectedIcon(new
ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_def.png");
contentPDFParametersUsedCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
contentPDFHeatMapCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_checked_dis.png");
contentPDFHeatMapCheckBox.setDisabledSelectedIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
contentPDFHeatMapCheckBox.setIcon(new ImageIcon(imgUrl));
imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
contentPDFHeatMapCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
```

```
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        contentPDFHeatMapCheckBox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        contentPDFHeatMapCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
        contentPDFClustersCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_dis.png");
        contentPDFClustersCheckBox.setDisabledSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
        contentPDFClustersCheckBox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        contentPDFClustersCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        contentPDFClustersCheckBox.setRolloverSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        contentPDFClustersCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
        contentPDFExpRatioCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_dis.png");
        contentPDFExpRatioCheckBox.setDisabledSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
        contentPDFExpRatioCheckBox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        contentPDFExpRatioCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        contentPDFExpRatioCheckBox.setRolloverSelectedIcon(new
        ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        contentPDFExpRatioCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_dis.png");
        contentPNGHeatMapCheckBox.setDisabledIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_dis.png");
        contentPNGHeatMapCheckBox.setDisabledSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
        contentPNGHeatMapCheckBox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        contentPNGHeatMapCheckBox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        contentPNGHeatMapCheckBox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        contentPNGHeatMapCheckBox.setSelectedIcon(new ImageIcon(imgUrl));
}else{
        contentXLSExpRatioCheckBox.setDisabledIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
        contentXLSExpRatioCheckBox.setDisabledSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_dis.png"));
        contentXLSExpRatioCheckBox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        contentXLSExpRatioCheckBox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        contentXLSExpRatioCheckBox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        contentXLSExpRatioCheckBox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
        contentXLSClustersCheckBox.setDisabledIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
        contentXLSClustersCheckBox.setDisabledSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_dis.png"));
        contentXLSClustersCheckBox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
```

```
contentXLSClustersCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentXLSClustersCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentXLSClustersCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentTXTExpRatioCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentTXTExpRatioCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentTXTExpRatioCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentTXTExpRatioCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentTXTExpRatioCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentTXTExpRatioCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentTXTClustersCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentTXTClustersCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentTXTClustersCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentTXTClustersCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentTXTClustersCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentTXTClustersCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentTXTParametersUsedCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentTXTParametersUsedCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentTXTParametersUsedCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentTXTParametersUsedCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentTXTParametersUsedCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentTXTParametersUsedCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentPDFParametersUsedCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentPDFParametersUsedCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentPDFParametersUsedCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentPDFParametersUsedCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
contentPDFParametersUsedCheckBox.setRolloverSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_hov.png"));
contentPDFParametersUsedCheckBox.setSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_def.png"));
contentPDFHeatMapCheckBox.setDisabledIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
contentPDFHeatMapCheckBox.setDisabledSelectedIcon(new
javax.swing.ImageIcon("src/img/box_checked_dis.png"));
contentPDFHeatMapCheckBox.setIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
contentPDFHeatMapCheckBox.setRolloverIcon(new
javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
```

```
        contentPDFHeatMapCheckBox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        contentPDFHeatMapCheckBox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
        contentPDFClustersCheckBox.setDisabledIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
        contentPDFClustersCheckBox.setDisabledSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_dis.png"));
        contentPDFClustersCheckBox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        contentPDFClustersCheckBox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        contentPDFClustersCheckBox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        contentPDFClustersCheckBox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
        contentPDFExpRatioCheckBox.setDisabledIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
        contentPDFExpRatioCheckBox.setDisabledSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_dis.png"));
        contentPDFExpRatioCheckBox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        contentPDFExpRatioCheckBox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        contentPDFExpRatioCheckBox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        contentPDFExpRatioCheckBox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
        contentPNGHeatMapCheckBox.setDisabledIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_dis.png"));
        contentPNGHeatMapCheckBox.setDisabledSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_dis.png"));
        contentPNGHeatMapCheckBox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        contentPNGHeatMapCheckBox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        contentPNGHeatMapCheckBox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        contentPNGHeatMapCheckBox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
}
contentXLSClustersCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentXLSClustersCheckBox.setText("Cluster Properties");
contentTXTExpRatioCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentTXTExpRatioCheckBox.setText("Expression Ratios");
contentTXTClustersCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentTXTClustersCheckBox.setText("Cluster Properties");
contentTXTClustersCheckBox.setEnabled(false);
contentTXTClustersCheckBox.setOpaque(false);
contentTXTParametersUsedCheckBox.setFont(new java.awt.Font
("Tahoma", 0, 10));
contentTXTParametersUsedCheckBox.setText("Parameters Used");
contentTXTParametersUsedCheckBox.setEnabled(false);
contentTXTParametersUsedCheckBox.setOpaque(false);
contentXLSExpRatioCheckBox.setEnabled(false);
contentXLSExpRatioCheckBox.setOpaque(false);
contentXLSClustersCheckBox.setEnabled(false);
contentXLSClustersCheckBox.setOpaque(false);
contentTXTExpRatioCheckBox.setEnabled(false);
contentTXTExpRatioCheckBox.setOpaque(false);
contentPDFParametersUsedCheckBox.setFont(new java.awt.Font
("Tahoma", 0, 10));
contentPDFParametersUsedCheckBox.setText("Parameters Used");
contentPDFParametersUsedCheckBox.setEnabled(false);
contentPDFParametersUsedCheckBox.setOpaque(false);
contentPDFHeatMapCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPDFHeatMapCheckBox.setText("Heatmap");
contentPDFHeatMapCheckBox.setEnabled(false);
contentPDFHeatMapCheckBox.setOpaque(false);
contentPDFClustersCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPDFClustersCheckBox.setText("Cluster Properties");
contentPDFClustersCheckBox.setEnabled(false);
contentPDFClustersCheckBox.setOpaque(false);
contentPDFExpRatioCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPDFExpRatioCheckBox.setText("Expression Ratios");
contentPDFExpRatioCheckBox.setEnabled(false);
contentPDFExpRatioCheckBox.setOpaque(false);
contentPNGHeatMapCheckBox.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPNGHeatMapCheckBox.setSelected(true);
contentPNGHeatMapCheckBox.setEnabled(false);
contentPNGHeatMapCheckBox.setOpaque(false);
exportPDFFileName.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
exportPDFFileName.setEnabled(false);
exportPDFFileName.setOpaque(false);
pdfLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
pdfLabel.setText(".pdf");
pdfLabel.setEnabled(false);
txtLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
txtLabel.setText(".txt");
txtLabel.setEnabled(false);
exportTextFileName.setHorizontalAlignment
(javax.swing.JTextField.RIGHT);
exportTextFileName.setEnabled(false);
exportTextFileName.setOpaque(false);
xlsLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
xlsLabel.setText(".xls");
xlsLabel.setEnabled(false);
exportExcelFileName.setHorizontalAlignment
(javax.swing.JTextField.RIGHT);
exportExcelFileName.setEnabled(false);
exportExcelFileName.setOpaque(false);
pngLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
pngLabel.setText(".png");
pngLabel.setEnabled(false);
exportImageFileName.setHorizontalAlignment
(javax.swing.JTextField.RIGHT);
exportImageFileName.setEnabled(false);
exportImageFileName.setOpaque(false);
contentPNGHeatmapLabel.setFont(new java.awt.Font("Tahoma", 0, 10));
contentPNGHeatmapLabel.setText("Heatmap");
contentPNGHeatmapLabel.setEnabled(false);
javax.swing.GroupLayout form4Layout = new javax.swing.GroupLayout(form4);
form4.setLayout(form4Layout);
form4Layout.setHorizontalGroup(form4Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING).addGroup(form4Layout.createSequentialGroup().addGap(
20, 20, 20)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form4Layout.createSequentialGroup().addGroup(form4Layout.createP
arallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(form4Layout.createSequentialGroup().addComponent(fileFormatsLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 83,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED).addComponent(fileNamesLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 127,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGroup(form4Layout.createParallelGro
up(javax.swing.GroupLayout.Alignment.TRAILING).addComponent(pngLabel).addGroup(
form4Layout.createSequentialGroup().addGroup(form4Layout.createParallelGroup(ja
vax.swing.GroupLayout.Alignment.TRAILING,
false).addComponent(exportToImageCheckbox,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
.addComponent(exportToExcelCheckbox, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(exportToTextCheckbox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addGroup(form4Layout.createSequentialGroup()
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(exportImageFileName, javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGroup(javax.swing.GroupLayout.Align
ment.TRAILING, form4Layout.createSequentialGroup().addGap(2, 2,
2).addComponent(exportExcelFileName, javax.swing.GroupLayout.PREFERRED_SIZE,
95,
javax.swing.GroupLayout.PREFERRED_SIZE))).addGroup(javax.swing.GroupLayout.Alig
nment.TRAILING, form4Layout.createSequentialGroup().addGap(2, 2,
2).addComponent(exportTextFileName, javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE))).addPreferredGap(javax.swing.LayoutSty
le.ComponentPlacement.RELATED).addGroup(form4Layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING).addComponent(txtLabel).addComponent(xlsLabe
l, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))))).addGap(18, 18, 18)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form4Layout.createSequentialGroup()
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRA
ILING,false).addComponent(contentXLSClustersCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE) .addComponent(contentTXTParametersUsedCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(contentTXTClustersCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)).addGap(18, 18,
18).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING).addComponent(contentTXTExpRatioCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(contentXLSExpRatioCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))).addComponent(contentsLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addGroup(form4Layout.createSequentialGroup().addGroup(form4Lay
out.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addGroup(for
m4Layout.createSequentialGroup().addComponent(contentPNGHeatMapCheckBox.addPref
erredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED).addComponent(conte
ntPNGHeatmapLabel)).addComponent(contentPDFParametersUsedCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 104,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(0, 0,
Short.MAX_VALUE))).addGap(33, 33, 33))
.addGroup(form4Layout.createSequentialGroup().addComponent(exportToPDFCheckbox,
javax.swing.GroupLayout.PREFERRED_SIZE, 83,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(exportPDFFileName, javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRA
ILING).addGroup(form4Layout.createSequentialGroup()    .addGap(52, 52,
52).addComponent(contentPDFHeatMapCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 104,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGroup(javax.swing.GroupLayout.Align
ment.LEADING, form4Layout.createSequentialGroup()
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(pdfLabel))).addGap(18, 18, 18)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addComponent(contentPDFClustersCheckBox).addComponent(contentPDFExpRatioC
heckBox, javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(0, 0,
Short.MAX_VALUE)))).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,form4La
yout.createSequentialGroup()
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(resetButtonForm4, javax.swing.GroupLayout.PREFERRED_SIZE,
111,javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutS
tyle.ComponentPlacement.RELATED).addComponent(exportButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(38, 38, 38))
.addGroup(form4Layout.createSequentialGroup().addContainerGap()
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form4Layout.createSequentialGroup()
.addComponent(destFolderLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 108,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED).addCompo
nent(destFolderPath, javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(chooseDestFolderButton)).addComponent(errorMessageForm4,
javax.swing.GroupLayout.PREFERRED_SIZE, 483,
javax.swing.GroupLayout.PREFERRED_SIZE)).addContainerGap()));

form4Layout.setVerticalGroup(form4Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING).addGroup(form4Layout.createSequentialGroup().addGap(30
, 30, 30)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(fileFormatsLabel).addComponent(fileNamesLabel).addComponent
(contentsLabel)).addGap(18, 18,
18).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false).addComponent(exportToImageCheckbox,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(contentPNGHeatMapCheckBox,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).add
Component(exportImageFileName).addComponent(pngLabel))
.addComponent(contentPNGHeatmapLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)).addGap(18, 18,
18).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING).addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE).addComponent(contentXLSClustersCheckBox).addComponent(conte
ntXLSExpRatioCheckBox)).addComponent(exportToExcelCheckbox,
javax.swing.GroupLayout.Alignment.TRAILING)).addGroup(javax.swing.GroupLayout.A
lignment.TRAILING,

form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE).add
Component(exportExcelFileName, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(xlsLabel)))
.addGap(18, 18, 18)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form4Layout.createSequentialGroup()
```

```
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(exportToTextCheckbox).addComponent(exportTextFileName,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(txtLabel))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(destFolderPath, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(chooseDestFolderButton).addComponent(destFolderLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)))
.addGroup(form4Layout.createSequentialGroup()
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(contentTXTParametersUsedCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 21,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(contentTXTExpRatioCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 21,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(contentTXTClustersCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(exportToPDFCheckbox).addComponent(exportPDFFileName,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(pdfLabel)
.addComponent(contentPDFClustersCheckBox).addComponent(contentPDFParametersUsed
CheckBox, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(contentPDFHeatMapCheckBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(contentPDFExpRatioCheckBox
, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(50, 50, 50)))
.addComponent(errorMessageForm4, javax.swing.GroupLayout.PREFERRED_SIZE, 17,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(1, 1, 1)
.addGroup(form4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(resetButtonForm4, javax.swing.GroupLayout.PREFERRED_SIZE,
23, javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(exportButton))
.addGap(58, 58, 58)));
form4.setBounds(10, 10, 510, 380);
contentPane.add(form4, javax.swing.JLayeredPane.DEFAULT_LAYER);
form3.setBackground(new java.awt.Color(255, 255, 255));
form3Instruction.setText("Set parameters for clustering
the microarray data.");
clusterButton.setText("Cluster");
clusterButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
      clusterButtonActionPerformed(evt);
}});
errorMessageForm3.setForeground(new java.awt.Color(255, 0, 0));
imgUrl = getClass().getResource("src/img/reset_def.png");
if(imgUrl !=null) resetButtonForm3.setIcon(new ImageIcon(imgUrl));
else resetButtonForm3.setIcon(new
javax.swing.ImageIcon("src/img/reset_def.png"));
resetButtonForm3.setText("Reset");
imgUrl = getClass().getResource("src/img/reset_hov.png");
if(imgUrl !=null) resetButtonForm3.setRolloverIcon(new ImageIcon(imgUrl));
```

```
else resetButtonForm3.setRolloverIcon(new
javax.swing.ImageIcon("src/img/reset_hov.png"));
resetButtonForm3.addActionListener
(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       resetButtonForm3ActionPerformed(evt);
}});
numberOfClustersLabel.setText("Number of clusters*");
clusterByGenesRadio.setSelected(true);
clusterByGenesRadio.setText("Genes");
clusterByGenesRadio.setOpaque(false);
clusterByGenesRadio.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       clusterByGenesRadioActionPerformed(evt);
}});
clusterBySamplesRadio.setText("Samples");
clusterBySamplesRadio.setOpaque(false);
clusterBySamplesRadio.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       clusterBySamplesRadioActionPerformed(evt);
}});
numOfClustText.setText("4");
distMetricLabel.setText("Distance Metric");
manhattanRadio.setText("Manhattan");
manhattanRadio.setOpaque(false);
euclidianRadio.setSelected(true);
euclidianRadio.setText("Euclidian");
euclidianRadio.setOpaque(false);
supnormRadio.setText("Supnorm");
supnormRadio.setOpaque(false);
initLearningRateLabel.setText("Initial Learning rate*");
initLearningRateText.setText("0.5");
maxIterLabel.setText("Number of Iterations*");
footnoteLabel2.setFont(new java.awt.Font("Tahoma", 0, 10));
footnoteLabel2.setForeground(new java.awt.Color(153, 153, 153));
footnoteLabel2.setText("* Required information");
javax.swing.GroupLayout form3Layout = new javax.swing.GroupLayout(form3);
form3.setLayout(form3Layout);
form3Layout.setHorizontalGroup(form3Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
form3Layout.createSequentialGroup()
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(resetButtonForm3, javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(clusterButton, javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(43, 43, 43))
.addGroup(form3Layout.createSequentialGroup().addGap(26, 26, 26)
.addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false).addComponent(form3Instruction,
javax.swing.GroupLayout.PREFERRED_SIZE, 386,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(form3Layout.createSequentialGroup().addGap(10, 10, 10)
.addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addComponent(errorMessageForm3, javax.swing.GroupLayout.PREFERRED_SIZE,
441,
javax.swing.GroupLayout.PREFERRED_SIZE).addGroup(form3Layout.createSequentialGr
oup().addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING, false).addComponent(maxIterLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, 116,
Short.MAX_VALUE).addComponent(initLearningRateLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(distMetricLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(numberOfClustersLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)).addGroup(form3Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING) .addGroup(form3Layout.createSequentialGroup().addGap(10,
10, 10)
.addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addComponent(manhattanRadio).addGroup(form3Layout.createSequentialGroup()
.addComponent(numOfClustText, javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18)
.addComponent(clusterByGenesRadio, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(clusterBySamplesRadio)).addComponent(euclidianRadio)
.addComponent(supnormRadio)))
.addGroup(form3Layout.createSequentialGroup().addPreferredGap(javax.swing.Layou
tStyle.ComponentPlacement.UNRELATED).addGroup(form3Layout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADING) .addComponent(initLearningRateText,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(maxIterText,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE))))).addComponent(footnoteLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 116,
javax.swing.GroupLayout.PREFERRED_SIZE)))).addContainerGap(33,
Short.MAX_VALUE)));

form3Layout.setVerticalGroup(form3Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING).addGroup(form3Layout.createSequentialGroup().addGap(25
, 25, 25).addComponent(form3Instruction,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(14, 14, 14)
.addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(numberOfClustersLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(numOfClustText,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(clusterByGenesRadio).addComponent(clusterBySamplesRadio))
.addGap(20, 20, 20)
.addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(manhattanRadio) .addComponent(distMetricLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(euclidianRadio).addPreferredGap(javax.swing.LayoutStyle.Component
Placement.UNRELATED).addComponent(supnormRadio)
.addGap(18, 18, 18)
.addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(initLearningRateText,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(initLearningRateLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(18, 18, 18)
.addGroup(form3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(maxIterLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(maxIterText,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(18, 18, 18)
.addComponent(footnoteLabel2).addPreferredGap(javax.swing.LayoutStyle.Component
Placement.UNRELATED).addComponent(errorMessageForm3,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED).addGroup(form3Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE).addComponent(resetButtonForm3,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(clusterButton))
.addContainerGap(30, Short.MAX_VALUE)));
form3.setBounds(10, 10, 510, 380);
contentPane.add(form3, javax.swing.JLayeredPane.DEFAULT_LAYER);
form2.setBackground(new java.awt.Color(255, 255, 255));
form2Instruction.setText("Choose preprocessing methods to use.");
preprocessButton.setText("Preprocess");
preprocessButton.addActionListener(new java.awt.event.ActionListener(){
public void actionPerformed(java.awt.event.ActionEvent evt) {
        preprocessButtonActionPerformed(evt);
}});
errorMessageForm2.setForeground(new java.awt.Color(255, 0, 0));
imgUrl = getClass().getResource("src/img/reset_def.png");
if(imgUrl !=null)resetButtonForm2.setIcon(new ImageIcon(imgUrl));
else resetButtonForm2.setIcon(new
javax.swing.ImageIcon("src/img/reset_def.png"));
resetButtonForm2.setText("Reset");
imgUrl = getClass().getResource("src/img/reset_hov.png");
if(imgUrl !=null)   resetButtonForm2.setRolloverIcon(new ImageIcon(imgUrl));
else resetButtonForm2.setRolloverIcon(new
javax.swing.ImageIcon("src/img/reset_hov.png"));
resetButtonForm2.addActionListener(new java.awt.event.ActionListener(){
public void actionPerformed(java.awt.event.ActionEvent evt) {
        resetButtonForm2ActionPerformed(evt);
}});
logTransCheckbox.setText("Logarithmic Transformation");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null){
        logTransCheckbox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        logTransCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        logTransCheckbox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        logTransCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
}else{
        logTransCheckbox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        logTransCheckbox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        logTransCheckbox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        logTransCheckbox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
}
logTransCheckbox.setOpaque(false);
zScoreTransCheckbox.setText("Z-Score Transformation");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null){
        zScoreTransCheckbox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        zScoreTransCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        zScoreTransCheckbox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        zScoreTransCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
}else{
        zScoreTransCheckbox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        zScoreTransCheckbox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        zScoreTransCheckbox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        zScoreTransCheckbox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
}
zScoreTransCheckbox.setOpaque(false);
medianCenteringCheckbox.setText("Median Centering");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null){
        medianCenteringCheckbox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        medianCenteringCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        medianCenteringCheckbox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        medianCenteringCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
}else{
        medianCenteringCheckbox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        medianCenteringCheckbox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        medianCenteringCheckbox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        medianCenteringCheckbox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
}
medianCenteringCheckbox.setOpaque(false);
scaleNormCheckbox.setText("Scale Normalization");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null){
        scaleNormCheckbox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        scaleNormCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        scaleNormCheckbox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        scaleNormCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
}else{
        scaleNormCheckbox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        scaleNormCheckbox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
        scaleNormCheckbox.setRolloverSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_hov.png"));
        scaleNormCheckbox.setSelectedIcon(new
        javax.swing.ImageIcon("src/img/box_checked_def.png"));
}
scaleNormCheckbox.setOpaque(false);
quantileNormCheckbox.setText("Quantile Normalization");
imgUrl = getClass().getResource("src/img/box_unchecked_def.png");
if(imgUrl !=null){
        quantileNormCheckbox.setIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_unchecked_hov.png");
        quantileNormCheckbox.setRolloverIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_hov.png");
        quantileNormCheckbox.setRolloverSelectedIcon(new ImageIcon(imgUrl));
        imgUrl = getClass().getResource("src/img/box_checked_def.png");
        quantileNormCheckbox.setSelectedIcon(new ImageIcon(imgUrl));
}else{
        quantileNormCheckbox.setIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_def.png"));
        quantileNormCheckbox.setRolloverIcon(new
        javax.swing.ImageIcon("src/img/box_unchecked_hov.png"));
```

```
       quantileNormCheckbox.setRolloverSelectedIcon(new
       javax.swing.ImageIcon("src/img/box_checked_hov.png"));
       quantileNormCheckbox.setSelectedIcon(new
       javax.swing.ImageIcon("src/img/box_checked_def.png"));
}
quantileNormCheckbox.setOpaque(false);
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) logTransHelp.setIcon(new ImageIcon(imgUrl));
else logTransHelp.setIcon(new javax.swing.ImageIcon("src/img/help_def.png"));
logTransHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) logTransHelp.setRolloverIcon(new ImageIcon(imgUrl));
else logTransHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
logTransHelp.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       logTransHelpActionPerformed(evt);
}});
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) medianCenteringHelp.setIcon(new ImageIcon(imgUrl));
else medianCenteringHelp.setIcon(new
javax.swing.ImageIcon("src/img/help_def.png"));
medianCenteringHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) medianCenteringHelp.setRolloverIcon(new ImageIcon(imgUrl));
else medianCenteringHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
medianCenteringHelp.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       medianCenteringHelpActionPerformed(evt);
}});
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) zScoreTransHelp.setIcon(new ImageIcon(imgUrl));
else zScoreTransHelp.setIcon(new
javax.swing.ImageIcon("src/img/help_def.png"));
zScoreTransHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) zScoreTransHelp.setRolloverIcon(new ImageIcon(imgUrl));
else zScoreTransHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
zScoreTransHelp.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       zScoreTransHelpActionPerformed(evt);
}});
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) scaleNormHelp.setIcon(new ImageIcon(imgUrl));
else scaleNormHelp.setIcon(new javax.swing.ImageIcon("src/img/help_def.png"));
scaleNormHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) scaleNormHelp.setRolloverIcon(new ImageIcon(imgUrl));
else scaleNormHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
scaleNormHelp.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       scaleNormHelpActionPerformed(evt);
}});
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) quantileNormHelp.setIcon(new ImageIcon(imgUrl));
else quantileNormHelp.setIcon(new
javax.swing.ImageIcon("src/img/help_def.png"));
quantileNormHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) quantileNormHelp.setRolloverIcon(new ImageIcon(imgUrl));
```

```
else quantileNormHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
quantileNormHelp.addActionListener(new java.awt.event.ActionListener(){
public void actionPerformed(java.awt.event.ActionEvent evt) {
       quantileNormHelpActionPerformed(evt);
}});
javax.swing.GroupLayout form2Layout = new javax.swing.GroupLayout(form2);

form2.setLayout(form2Layout);
form2Layout.setHorizontalGroup(
form2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addG
roup(javax.swing.GroupLayout.Alignment.TRAILING,
form2Layout.createSequentialGroup().addContainerGap(javax.swing.GroupLayout.DEF
AULT_SIZE, Short.MAX_VALUE).addComponent(resetButtonForm2,
javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED).addComponent(preprocessButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(43, 43, 43))
.addGroup(form2Layout.createSequentialGroup().addGroup(form2Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(form2Layout.createSequentialGroup().addGap(36, 36, 36)
.addGroup(form2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addComponent(errorMessageForm2, javax.swing.GroupLayout.PREFERRED_SIZE,
441, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(form2Layout.createSequentialGroup()
.addGroup(form2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addComponent(logTransCheckbox).addComponent(zScoreTransCheckbox).addCompo
nent(scaleNormCheckbox).addComponent(quantileNormCheckbox).addComponent(medianC
enteringCheckbox, javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE)).addPreferredGap(javax.swing.LayoutStyl
e.ComponentPlacement.UNRELATED).addGroup(form2Layout.createParallelGroup(javax.
swing.GroupLayout.Alignment.TRAILING, false) .addComponent(scaleNormHelp,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 1, Short.MAX_VALUE)
.addComponent(medianCenteringHelp, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 1, Short.MAX_VALUE)
.addComponent(zScoreTransHelp, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
.addComponent(logTransHelp, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(quantileNormHelp, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE))))).addGroup(form2Layout.createSequentialGroup() .addGap(28,
28, 28).addComponent(form2Instruction, javax.swing.GroupLayout.PREFERRED_SIZE,
386, javax.swing.GroupLayout.PREFERRED_SIZE))).addContainerGap(33,
Short.MAX_VALUE)));

form2Layout.setVerticalGroup(form2Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING).addGroup(form2Layout.createSequentialGroup().addGap(42
, 42, 42).addComponent(form2Instruction,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(28, 28, 28)
.addGroup(form2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(logTransCheckbox).addComponent(logTransHelp)).addPreferredG
ap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED).addGroup(form2Layout.c
reateParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addComponent(zSco
reTransCheckbox).addComponent(zScoreTransHelp)).addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.UNRELATED).addGroup(form2Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.BASELINE).addComponent(medianCenteringCheckb
ox, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(medianCenteringHelp)).addP
referredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(form2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(scaleNormCheckbox, javax.swing.GroupLayout.PREFERRED_SIZE,
27, javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(scaleNormHelp))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(form2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(quantileNormCheckbox).addComponent(quantileNormHelp)).addGa
p(18, 18, 18).addComponent(errorMessageForm2,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.UNRELATED).addGroup(form2Layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.BASELINE).addComponent(resetButtonForm2,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(preprocessButton))
.addContainerGap(43, Short.MAX_VALUE)));

form2.setBounds(10, 10, 510, 380);
contentPane.add(form2, javax.swing.JLayeredPane.DEFAULT_LAYER);
form1.setBackground(new java.awt.Color(255, 255, 255));
form1Instruction.setText("Choose files to process.");
dataFileLabel.setText("Data file*");
sampleDescriptionLabel.setText("Sample description file");
geneDescriptionLabel.setText("Gene description file");
dataFilePath.setEditable(false);
dataFilePath.setEnabled(false);
dataFilePath.setOpaque(false);
geneFilePath.setEditable(false);
geneFilePath.setEnabled(false);
geneFilePath.setOpaque(false);
sampleFilePath.setEditable(false);
sampleFilePath.setEnabled(false);
sampleFilePath.setOpaque(false);
chooseDataFileButton.setBackground(new java.awt.Color(255, 255, 255));
imgUrl = getClass().getResource("src/img/folder_def.png");
if(imgUrl !=null) chooseDataFileButton.setIcon(new ImageIcon(imgUrl));
else chooseDataFileButton.setIcon(new
javax.swing.ImageIcon("src/img/folder_def.png"));
chooseDataFileButton.setText("Choose");
imgUrl = getClass().getResource("src/img/folder_hov.png");
if(imgUrl !=null) chooseDataFileButton.setRolloverIcon(new ImageIcon(imgUrl));
else chooseDataFileButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/folder_hov.png"));
chooseDataFileButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       chooseDataFileButtonActionPerformed(evt);
}});
chooseGeneFileButton.setBackground(new java.awt.Color(255, 255, 255));
imgUrl = getClass().getResource("src/img/folder_def.png");
if(imgUrl !=null) chooseGeneFileButton.setIcon(new ImageIcon(imgUrl));
else chooseGeneFileButton.setIcon(new
javax.swing.ImageIcon("src/img/folder_def.png"));
chooseGeneFileButton.setText("Choose");
imgUrl = getClass().getResource("src/img/folder_hov.png");
if(imgUrl !=null) chooseGeneFileButton.setRolloverIcon(new ImageIcon(imgUrl));
else chooseGeneFileButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/folder_hov.png"));
chooseGeneFileButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       chooseGeneFileButtonActionPerformed(evt);
}});
chooseSampleFileButton.setBackground
(new java.awt.Color(255, 255, 255));
imgUrl = getClass().getResource("src/img/folder_def.png");
if(imgUrl !=null) chooseSampleFileButton.setIcon(new ImageIcon(imgUrl));
```

```
else chooseSampleFileButton.setIcon(new
javax.swing.ImageIcon("src/img/folder_def.png"));
chooseSampleFileButton.setText("Choose");
imgUrl = getClass().getResource("src/img/folder_hov.png");
if(imgUrl !=null) chooseSampleFileButton.setRolloverIcon(new
ImageIcon(imgUrl));
else chooseSampleFileButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/folder_hov.png"));
chooseSampleFileButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       chooseSampleFileButtonActionPerformed(evt);
}});
dataFileHelp.setBackground(new java.awt.Color(255, 255, 255));
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) dataFileHelp.setIcon(new ImageIcon(imgUrl));
else dataFileHelp.setIcon(new javax.swing.ImageIcon("src/img/help_def.png"));
dataFileHelp.setBorderPainted(false);
dataFileHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) dataFileHelp.setRolloverIcon(new ImageIcon(imgUrl));
else dataFileHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
dataFileHelp.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       dataFileHelpActionPerformed(evt);
}});
geneFileHelp.setBackground(new java.awt.Color(255, 255, 255));
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) geneFileHelp.setIcon(new ImageIcon(imgUrl));
else geneFileHelp.setIcon(new javax.swing.ImageIcon("src/img/help_def.png"));
geneFileHelp.setBorderPainted(false);
geneFileHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) geneFileHelp.setRolloverIcon(new ImageIcon(imgUrl));
else geneFileHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
geneFileHelp.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       geneFileHelpActionPerformed(evt);
}});
sampleFileHelp.setBackground(new java.awt.Color(255, 255, 255));
imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) sampleFileHelp.setIcon(new ImageIcon(imgUrl));
else sampleFileHelp.setIcon(new javax.swing.ImageIcon("src/img/help_def.png"));
sampleFileHelp.setBorderPainted(false);
sampleFileHelp.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl !=null) sampleFileHelp.setRolloverIcon(new ImageIcon(imgUrl));
else sampleFileHelp.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
sampleFileHelp.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       sampleFileHelpActionPerformed(evt);
}});
uploadFileButton.setText("Submit");
uploadFileButton.addActionListener(new java.awt.event.ActionListener(){
public void actionPerformed(java.awt.event.ActionEvent evt) {
       uploadFileButtonActionPerformed(evt);
}});
footnoteLabel1.setFont(new java.awt.Font("Tahoma", 0, 10));
footnoteLabel1.setForeground(new java.awt.Color(153, 153, 153));
footnoteLabel1.setText("* Required information");
errorMessageForm1.setForeground(new java.awt.Color(255, 0, 0));
```

```
imgUrl = getClass().getResource("src/img/reset_def.png");
if(imgUrl !=null) resetButtonForm1.setIcon(new ImageIcon(imgUrl));
else resetButtonForm1.setIcon(new
javax.swing.ImageIcon("src/img/reset_def.png"));
resetButtonForm1.setText("Reset");
imgUrl = getClass().getResource("src/img/reset_hov.png");
if(imgUrl !=null) resetButtonForm1.setRolloverIcon(new ImageIcon(imgUrl));
else resetButtonForm1.setRolloverIcon(new
javax.swing.ImageIcon("src/img/reset_hov.png"));
resetButtonForm1.addActionListener(new java.awt.event.ActionListener(){
public void actionPerformed(java.awt.event.ActionEvent evt) {
       resetButtonForm1ActionPerformed(evt);
}});
javax.swing.GroupLayout form1Layout = new javax.swing.GroupLayout(form1);
form1.setLayout(form1Layout);
form1Layout.setHorizontalGroup(form1Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING).addGroup(form1Layout.createSequentialGroup().addGap(
26, 26, 26)
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form1Layout.createSequentialGroup()
.addComponent(footnoteLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 118,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0, 0,
Short.MAX_VALUE)).addGroup(form1Layout.createSequentialGroup()
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form1Layout.createSequentialGroup()
.addComponent(form1Instruction, javax.swing.GroupLayout.PREFERRED_SIZE, 386,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0, 0,
Short.MAX_VALUE)).addGroup(form1Layout.createSequentialGroup()
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form1Layout.createSequentialGroup()
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addComponent(sampleDescriptionLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE) .addComponent(geneDescriptionLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE) .addComponent(dataFileLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)).addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED).addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING, false).addComponent(dataFilePath,
javax.swing.GroupLayout.PREFERRED_SIZE, 197,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(sampleFilePath,
javax.swing.GroupLayout.PREFERRED_SIZE, 197,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(geneFilePath,
javax.swing.GroupLayout.PREFERRED_SIZE, 197,
javax.swing.GroupLayout.PREFERRED_SIZE))).addGroup(form1Layout.createSequential
Group().addGap(0, 0, Short.MAX_VALUE) .addComponent(resetButtonForm1,
javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE)))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.TRAILING).addGroup(form1Layout.createSequentialGroup().addComponent(chooseGen
eFileButton).addGap(2, 2, 2) .addComponent(geneFileHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGroup(form1Layout.createSequentialGroup()
.addComponent(chooseDataFileButton).addGap(2, 2, 2) .addComponent(dataFileHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)))
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addComponent(uploadFileButton, javax.swing.GroupLayout.PREFERRED_SIZE,
112, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
form1Layout.createSequentialGroup()
.addComponent(chooseSampleFileButton).addGap(2, 2, 2)
.addComponent(sampleFileHelp, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(8, 8, 8)))))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
form1Layout.createSequentialGroup().addGap(0, 0, Short.MAX_VALUE)
.addComponent(errorMessageForm1, javax.swing.GroupLayout.PREFERRED_SIZE, 441,
javax.swing.GroupLayout.PREFERRED_SIZE))).addGap(43, 43, 43)))));

form1Layout.setVerticalGroup(form1Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING).addGroup(form1Layout.createSequentialGroup().addGap(40
, 40, 40).addComponent(form1Instruction,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(30, 30, 30)
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.BASELINE).addComponent(dataFileLabel, javax.swing.GroupLayout.PREFERRED_SIZE,
23, javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(dataFilePath,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(chooseDataFileButton)).addComponent(dataFileHelp,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(17, 17, 17)
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(form1Layout.createSequentialGroup()
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(geneFilePath, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(chooseGeneFileButton).addComponent(geneFileHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(10, 10, 10))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
form1Layout.createSequentialGroup().addComponent(geneDescriptionLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18)))
.addGroup(form1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE).addComponent(sampleDescriptionLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(sampleFilePath,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(chooseSampleFileButton).addComponent(sampleFileHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(18, 18, 18)
.addComponent(footnoteLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(errorMessageForm1, javax.swing.GroupLayout.PREFERRED_SIZE, 17,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.UNRELATED).addGroup(form1Layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.BASELINE).addComponent(uploadFileButton).addComponen
t(resetButtonForm1, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(60, 60, 60)));
form1.setBounds(10, 10, 510, 380);
contentPane.add(form1, javax.swing.JLayeredPane.DEFAULT_LAYER);
javax.swing.GroupLayout mainContentPanelLayout = new
javax.swing.GroupLayout(mainContentPanel);
mainContentPanel.setLayout(mainContentPanelLayout);
```

```
mainContentPanelLayout.setHorizontalGroup(mainContentPanelLayout.createParallel
Group(javax.swing.GroupLayout.Alignment.LEADING).addGroup(mainContentPanelLayou
t.createSequentialGroup().addContainerGap().addGroup(mainContentPanelLayout.cre
ateParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addComponent(progre
ssPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(contentPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 525,
javax.swing.GroupLayout.PREFERRED_SIZE)).addPreferredGap(javax.swing.LayoutStyl
e.ComponentPlacement.RELATED).addGroup(mainContentPanelLayout.createParallelGro
up(javax.swing.GroupLayout.Alignment.LEADING).addComponent(navigationButtonsPan
el, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(jPanel3,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)).addContainerGap()));

mainContentPanelLayout.setVerticalGroup(mainContentPanelLayout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING).addGroup(mainContentPanelLayout.
createSequentialGroup().addContainerGap().addGroup(mainContentPanelLayout.creat
eParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addGroup(mainContentP
anelLayout.createSequentialGroup().addGap(0, 0,
Short.MAX_VALUE).addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)).addComponent(contentPane)).addPreferre
dGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED).addGroup(mainContentPa
nelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addGro
up(mainContentPanelLayout.createSequentialGroup().addComponent(progressPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(43, 43,
43)).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
mainContentPanelLayout.createSequentialGroup().addComponent(navigationButtonsPa
nel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(54, 54, 54)))));

imgUrl = getClass().getResource("src/img/help_def.png");
if(imgUrl !=null) mainHelpButton.setIcon(new ImageIcon(imgUrl));
else mainHelpButton.setIcon(new javax.swing.ImageIcon("src/img/help_def.png"));
mainHelpButton.setText("Help");
mainHelpButton.setContentAreaFilled(false);
imgUrl = getClass().getResource("src/img/help_hov.png");
if(imgUrl!=null) mainHelpButton.setRolloverIcon(new ImageIcon(imgUrl));
else mainHelpButton.setRolloverIcon(new
javax.swing.ImageIcon("src/img/help_hov.png"));
mainHelpButton.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
       mainHelpButtonActionPerformed(evt);
}});
javax.swing.GroupLayout backgroundPanelLayout = new
javax.swing.GroupLayout(backgroundPanel);
backgroundPanel.setLayout(backgroundPanelLayout);

backgroundPanelLayout.setHorizontalGroup(
backgroundPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
backgroundPanelLayout.createSequentialGroup().addGap(16, 16,
16).addComponent(DataIcon, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED).addComponent(PreprocessIcon).addPreferredGap(javax
.swing.LayoutStyle.ComponentPlacement.RELATED).addComponent(ClusterIcon).addPre
ferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED).addComponent(Visu
alizeIcon).addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE,
        Short.MAX_VALUE).addComponent(mainHelpButton).addContainerGap()).addComponent(B
        anner, javax.swing.GroupLayout.Alignment.TRAILING,
        javax.swing.GroupLayout.DEFAULT_SIZE, 846,
        Short.MAX_VALUE).addGroup(backgroundPanelLayout.createParallelGroup(javax.swing
        .GroupLayout.Alignment.LEADING).addGroup(backgroundPanelLayout.createSequential
        Group().addContainerGap().addComponent(mainContentPanel,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap(10,
        Short.MAX_VALUE))));

        backgroundPanelLayout.setVerticalGroup(backgroundPanelLayout.createParallelGrou
        p(javax.swing.GroupLayout.Alignment.LEADING).addGroup(backgroundPanelLayout.cre
        ateSequentialGroup
        .addComponent(Banner, javax.swing.GroupLayout.PREFERRED_SIZE, 77,
        javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle
        .ComponentPlacement.RELATED).addGroup(backgroundPanelLayout.createParallelGroup
        (javax.swing.GroupLayout.Alignment.LEADING).addComponent(mainHelpButton).addCom
        ponent(PreprocessIcon, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
        javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(ClusterIcon,
        javax.swing.GroupLayout.PREFERRED_SIZE, 24,
        javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(VisualizeIcon,
        javax.swing.GroupLayout.PREFERRED_SIZE, 24,
        javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(DataIcon,
        javax.swing.GroupLayout.PREFERRED_SIZE, 24,
        javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(0, 0, Short.MAX_VALUE))
        .addGroup(backgroundPanelLayout.createParallelGroup(javax.swing.GroupLayout.Ali
        gnment.LEADING).addGroup(backgroundPanelLayout.createSequentialGroup().addGap(1
        13, 113, 113).addComponent(mainContentPanel,
        javax.swing.GroupLayout.PREFERRED_SIZE, 478,
        javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap(javax.swing.GroupLayout
        .DEFAULT_SIZE, Short.MAX_VALUE))));
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Al
        ignment.LEADING).addComponent(backgroundPanel,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
        Short.MAX_VALUE));
        layout.setVerticalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alig
        nment.LEADING).addComponent(backgroundPanel,
        javax.swing.GroupLayout.DEFAULT_SIZE, 591, Short.MAX_VALUE));

        pack();
        this.setLocationRelativeTo(null);
}

private void viewDataButtonActionPerformed(java.awt.event.ActionEvent evt) {
        results.rgui.tabbedPane.setSelectedIndex(0);
        results.rgui.setVisible(true);
}
private void backButtonActionPerformed(java.awt.event.ActionEvent evt) {
        status.setText("\t");
        progressbar.setValue(0);
        if(form2.isVisible()){
                form2.setVisible(false);
                form1.setVisible(true);
                backButton.setEnabled(false);
                nextButton.setText("Next");
                nextButton.setEnabled(true);
                PreprocessIcon.setEnabled(false);
                DataIcon.setEnabled(true);
                stepTitle.setText(formTitles[0]);
                stepDescription.setText(formDescriptions[0]);
                URL imgUrl = getClass().getResource(formImages[0]);
                if(imgUrl !=null) stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
                else stepBigImageLabel.setIcon(new ImageIcon(formImages[0]));
        }else if(form3.isVisible()){
                form3.setVisible(false);
                form2.setVisible(true);
                backButton.setEnabled(true);
                nextButton.setText("Next");
                nextButton.setEnabled(true);
                ClusterIcon.setEnabled(false);
                PreprocessIcon.setEnabled(true);
                stepTitle.setText(formTitles[1]);
                stepDescription.setText(formDescriptions[1]);
                URL imgUrl = getClass().getResource(formImages[1]);
                if(imgUrl !=null) stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
                else stepBigImageLabel.setIcon(new ImageIcon(formImages[1]));
        }else if(form4.isVisible()){
                form4.setVisible(false);
                form3.setVisible(true);
                backButton.setEnabled(true);
                nextButton.setText("Next");
                nextButton.setEnabled(true);
                VisualizeIcon.setEnabled(false);
                ClusterIcon.setEnabled(true);
                stepTitle.setText(formTitles[2]);
                stepDescription.setText(formDescriptions[2]);
                URL imgUrl = getClass().getResource(formImages[2]);
                if(imgUrl !=null)stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
                else stepBigImageLabel.setIcon(new ImageIcon(formImages[2]));
                viewHeatMapButton.setVisible(false);
                viewDataButton.setVisible(false);
        }
}

private void chooseDataFileButtonActionPerformed(java.awt.event.ActionEvent evt){
        int returnVal = fileChooser.showOpenDialog( this );
        if (returnVal == JFileChooser.APPROVE_OPTION) {
                dataFile = fileChooser.getSelectedFile();
                dataFilePath.setText(dataFile.getAbsolutePath());
        }
}

private void mainHelpButtonActionPerformed(java.awt.event.ActionEvent evt) {
        helpWindow.setVisible(true);
}

private void dataFileHelpActionPerformed(java.awt.event.ActionEvent evt) {
        helpWindow.dispose();
        helpWindow = new HelpGUI(2);
}

private void geneFileHelpActionPerformed(java.awt.event.ActionEvent evt) {
        helpWindow.dispose();
        helpWindow = new HelpGUI(3);
}

private void sampleFileHelpActionPerformed(java.awt.event.ActionEvent evt) {
        helpWindow.dispose();
        helpWindow = new HelpGUI(3);
}

private void uploadFileButtonActionPerformed(java.awt.event.ActionEvent evt){
        resetData();
        backgroundPanel.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
```

```
       nextButton.setEnabled(false);
       uploadFile = new uploadFileWorker();
       uploadFile.execute();
}

private void resetButtonForm1ActionPerformed(java.awt.event.ActionEvent evt){
       dataFilePath.setText("");
       geneFilePath.setText("");
       sampleFilePath.setText("");
       errorMessageForm1.setText("");
       errorMessageForm1.setVisible(false);
       status.setText("\t");
       progressbar.setValue(0);
       if(uploadFile != null )uploadFile.cancel(true);
}

private void chooseGeneFileButtonActionPerformed(java.awt.event.ActionEvent evt) {
       int returnVal = fileChooser.showOpenDialog( this );
       if (returnVal == JFileChooser.APPROVE_OPTION) {
              geneFile = fileChooser.getSelectedFile();
              geneFilePath.setText(geneFile.getAbsolutePath());
       }
}

private void chooseSampleFileButtonActionPerformed(java.awt.event.ActionEvent evt) {
       int returnVal = fileChooser.showOpenDialog( this );
       if (returnVal == JFileChooser.APPROVE_OPTION) {
              sampleFile = fileChooser.getSelectedFile();
              sampleFilePath.setText(sampleFile.getAbsolutePath());
       }
}

private void nextButtonActionPerformed(java.awt.event.ActionEvent evt) {
       status.setText("\t");
       progressbar.setValue(0);
       if(form1.isVisible()){
              form1.setVisible(false);
              form2.setVisible(true);
              backButton.setEnabled(true);
              nextButton.setText("Skip");
              nextButton.setEnabled(true);
              DataIcon.setEnabled(false);
              PreprocessIcon.setEnabled(true);
              stepTitle.setText(formTitles[1]);
              stepDescription.setText(formDescriptions[1]);
              URL imgUrl = getClass().getResource(formImages[1]);
              if(imgUrl !=null) stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
              else stepBigImageLabel.setIcon(new ImageIcon(formImages[1]));
       }else if(form2.isVisible()){
              form2.setVisible(false);
              form3.setVisible(true);
              backButton.setEnabled(true);
              nextButton.setEnabled(false);
              nextButton.setText("Next");
              PreprocessIcon.setEnabled(false);
              ClusterIcon.setEnabled(true);
              stepTitle.setText(formTitles[2]);
              stepDescription.setText(formDescriptions[2]);
              URL imgUrl = getClass().getResource(formImages[2]);
              if(imgUrl !=null) stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
              else stepBigImageLabel.setIcon(new ImageIcon(formImages[2]));
              if(clusterByGenesRadio.isSelected()) maxIterText.setText(""+
              fileParser.dataRowLabels.length*5);
```

```
        else maxIterText.setText(""+ fileParser.dataColLabels.length*5);
    }else if(form3.isVisible()){
        form3.setVisible(false);
        form4.setVisible(true);
        backButton.setEnabled(true);
        nextButton.setText("New");
        nextButton.setEnabled(true);
        ClusterIcon.setEnabled(false);
        VisualizeIcon.setEnabled(true);
        stepTitle.setText(formTitles[3]);
        stepDescription.setText(formDescriptions[3]);
        URL imgUrl = getClass().getResource(formImages[3]);
        if(imgUrl !=null) stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
        else stepBigImageLabel.setIcon(new ImageIcon(formImages[3]));
        viewHeatMapButton.setVisible(true);
        viewDataButton.setVisible(true);
        if(isNewCluster){
            heatmapPanel = new HeatMap(data,clusterAssignments,
            som.isByCols,true,Gradient.
            GRADIENT_GREEN_YELLOW_ORANGE_RED);
            displayDataResults(heatmapPanel);
            isNewCluster =false;
        }
    }else if(form4.isVisible()){
        form4.setVisible(false);
        form1.setVisible(true);
        backButton.setEnabled(false);
        nextButton.setEnabled(false);
        nextButton.setText("Next");
        VisualizeIcon.setEnabled(false);
        DataIcon.setEnabled(true);
        stepTitle.setText(formTitles[0]);
        stepDescription.setText(formDescriptions[0]);
        URL imgUrl = getClass().getResource(formImages[0]);
        if(imgUrl !=null) stepBigImageLabel.setIcon(new ImageIcon(imgUrl));
        else stepBigImageLabel.setIcon(new ImageIcon(formImages[0]));
        viewHeatMapButton.setVisible(false);
        viewDataButton.setVisible(false);
    }
}

private void clusterButtonActionPerformed(java.awt.event.ActionEvent evt) {
    backgroundPanel.setCursor(Cursor.getPredefinedCursor
    (Cursor.WAIT_CURSOR));
    nextButton.setEnabled(false);
    errorMessageForm3.setText("");
    cluster = new clusterWorker();
    cluster.execute();
}

private void resetButtonForm3ActionPerformed(java.awt.event.ActionEvent evt){
    clusterByGenesRadio.setSelected(true);
    euclidianRadio.setSelected(true);
    numOfClustText.setText("4");
    initLearningRateText.setText("0.5");
    maxIterText.setText(""+ fileParser.dataRowLabels.length*5);
    errorMessageForm3.setVisible(false);
    status.setText("\t");
    progressbar.setValue(0);
    if(cluster != null ) cluster.cancel(true);
}

private void clusterByGenesRadioActionPerformed(java.awt.event.ActionEvent evt) {
       maxIterText.setText(""+ fileParser.dataRowLabels.length*5);
}

private void clusterBySamplesRadioActionPerformed(java.awt.event.ActionEvent evt) {
       maxIterText.setText(""+ fileParser.dataColLabels.length*5);
}

private void exportToImageCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
       exportImageFileName.setEnabled(exportToImageCheckbox.isSelected());
       contentPNGHeatmapLabel.setEnabled(exportToImageCheckbox.isSelected());
       pngLabel.setEnabled(exportToImageCheckbox.isSelected());
}

private void chooseDestFolderButtonActionPerformed(java.awt.event.ActionEvent evt) {
       folderChooser.setCurrentDirectory(new java.io.File("."));
       folderChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
       folderChooser.setAcceptAllFileFilterUsed(false);
       if (folderChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
              destFolderPath.setText(""+folderChooser.getSelectedFile());
}

private void resetButtonForm4ActionPerformed(java.awt.event.ActionEvent evt){
       contentXLSClustersCheckBox.setSelected(false);
       contentXLSExpRatioCheckBox.setSelected(false);
       contentPDFClustersCheckBox.setSelected(false);
       contentPDFExpRatioCheckBox.setSelected(false);
       contentPDFHeatMapCheckBox.setSelected(false);
       contentPDFParametersUsedCheckBox.setSelected(false);
       contentTXTClustersCheckBox.setSelected(false);
       contentTXTExpRatioCheckBox.setSelected(false);
       contentTXTParametersUsedCheckBox.setSelected(false);
       contentXLSClustersCheckBox.setEnabled(false);
       contentXLSExpRatioCheckBox.setEnabled(false);
       contentPDFClustersCheckBox.setEnabled(false);
       contentPDFExpRatioCheckBox.setEnabled(false);
       contentPDFHeatMapCheckBox.setEnabled(false);
       contentPDFParametersUsedCheckBox.setEnabled(false);
       contentPNGHeatmapLabel.setEnabled(false);
       contentTXTClustersCheckBox.setEnabled(false);
       contentTXTExpRatioCheckBox.setEnabled(false);
       contentTXTParametersUsedCheckBox.setEnabled(false);
       exportExcelFileName.setText("");
       exportImageFileName.setText("");
       exportPDFFileName.setText("");
       exportTextFileName.setText("");
       exportExcelFileName.setEnabled(false);
       exportImageFileName.setEnabled(false);
       exportPDFFileName.setEnabled(false);
       exportTextFileName.setEnabled(false);
       pngLabel.setEnabled(false);
       txtLabel.setEnabled(false);
       pdfLabel.setEnabled(false);
       xlsLabel.setEnabled(false);
       exportToExcelCheckbox.setSelected(false);
       exportToImageCheckbox.setSelected(false);
       exportToPDFCheckbox.setSelected(false);
       exportToTextCheckbox.setSelected(false);
       destFolderPath.setText("");
       errorMessageForm4.setText("");
       status.setText("\t");
       progressbar.setValue(0);
       if(exportFile != null) exportFile.cancel(true);
}
```

```
private void exportButtonActionPerformed(java.awt.event.ActionEvent evt) {
       backgroundPanel.setCursor(Cursor.getPredefinedCursor
       (Cursor.WAIT_CURSOR));
       exportFile = new exportFileWorker();
       exportFile.execute();
}

private void exportToTextCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
       contentTXTClustersCheckBox.setEnabled
       (exportToTextCheckbox.isSelected());
       contentTXTExpRatioCheckBox.setEnabled
       (exportToTextCheckbox.isSelected());
       contentTXTParametersUsedCheckBox.setEnabled
       (exportToTextCheckbox.isSelected());
       exportTextFileName.setEnabled(exportToTextCheckbox.isSelected());
       txtLabel.setEnabled(exportToTextCheckbox.isSelected());
}

private void exportToPDFCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
       contentPDFClustersCheckBox.setEnabled
       (exportToPDFCheckbox.isSelected());
       contentPDFExpRatioCheckBox.setEnabled
       (exportToPDFCheckbox.isSelected());
       contentPDFHeatMapCheckBox.setEnabled
       (exportToPDFCheckbox.isSelected());
       contentPDFParametersUsedCheckBox.setEnabled
       (exportToPDFCheckbox.isSelected());
       exportPDFFileName.setEnabled(exportToPDFCheckbox.isSelected());
       pdfLabel.setEnabled(exportToPDFCheckbox.isSelected());
}

private void exportToExcelCheckboxActionPerformed(java.awt.event.ActionEvent evt) {
       contentXLSClustersCheckBox.setEnabled
       (exportToExcelCheckbox.isSelected());
       contentXLSExpRatioCheckBox.setEnabled
       (exportToExcelCheckbox.isSelected());
       exportExcelFileName.setEnabled(exportToExcelCheckbox.isSelected());
       xlsLabel.setEnabled(exportToExcelCheckbox.isSelected());
}

private void viewHeatMapButtonActionPerformed(java.awt.event.ActionEvent evt){
       results.rgui.tabbedPane.setSelectedIndex(2);
       results.rgui.setVisible(true);
}
private void logTransHelpActionPerformed(java.awt.event.ActionEvent evt) {
       helpWindow.dispose();
       helpWindow = new HelpGUI(4);
}

private void zScoreTransHelpActionPerformed(java.awt.event.ActionEvent evt) {
       helpWindow.dispose();
       helpWindow = new HelpGUI(5);
}

private void medianCenteringHelpActionPerformed(java.awt.event.ActionEvent evt) {
       helpWindow.dispose();
       helpWindow = new HelpGUI(5);
}

private void scaleNormHelpActionPerformed(java.awt.event.ActionEvent evt) {
       helpWindow.dispose();
       helpWindow = new HelpGUI(5);
}

private void quantileNormHelpActionPerformed(java.awt.event.ActionEvent evt){
       helpWindow.dispose();
       helpWindow = new HelpGUI(5);
}

private void resetButtonForm2ActionPerformed(java.awt.event.ActionEvent evt){
       logTransCheckbox.setSelected(false);
       zScoreTransCheckbox.setSelected(false);
       medianCenteringCheckbox.setSelected(false);
       scaleNormCheckbox.setSelected(false);
       quantileNormCheckbox.setSelected(false);
       errorMessageForm2.setText("");
       errorMessageForm2.setVisible(false);
       status.setText("\t");
       progressbar.setValue(0);
       if(preprocess != null) preprocess.cancel(true);
}

private void preprocessButtonActionPerformed(java.awt.event.ActionEvent evt){
       backgroundPanel.setCursor(Cursor.getPredefinedCursor
       (Cursor.WAIT_CURSOR));
       errorMessageForm2.setText("");
       preprocess = new preprocessWorker();
       preprocess.execute();
}

public static void main(String args[]) {
       try { // Set System L&F
              javax.swing.UIManager.setLookAndFeel(javax.swing.UIManager.getSystemLookA
              ndFeelClassName());
       }catch (javax.swing.UnsupportedLookAndFeelException | ClassNotFoundException |
       InstantiationException | IllegalAccessException e) {}
       /* Create and display the form */
       java.awt.EventQueue.invokeLater(new Runnable() {
              public void run() {
              new UserInterface().setVisible(true);
              }
       });
}

private javax.swing.JLabel Banner;
private javax.swing.JLabel ClusterIcon;
private javax.swing.JLabel DataIcon;
private javax.swing.JLabel PreprocessIcon;
private javax.swing.JLabel VisualizeIcon;
private javax.swing.JButton backButton;
private javax.swing.JPanel backgroundPanel;
private javax.swing.JButton chooseDataFileButton;
private javax.swing.JButton chooseDestFolderButton;
private javax.swing.JButton chooseGeneFileButton;
private javax.swing.JButton chooseSampleFileButton;
private javax.swing.JButton clusterButton;
private javax.swing.JRadioButton clusterByGenesRadio;
private javax.swing.JRadioButton clusterBySamplesRadio;
private javax.swing.JCheckBox contentPDFClustersCheckBox;
private javax.swing.JCheckBox contentPDFExpRatioCheckBox;
private javax.swing.JCheckBox contentPDFHeatMapCheckBox;
private javax.swing.JCheckBox contentPDFParametersUsedCheckBox;
private javax.swing.JCheckBox contentPNGHeatMapCheckBox;
private javax.swing.JLabel contentPNGHeatmapLabel;
private javax.swing.JLayeredPane contentPane;
private javax.swing.JCheckBox contentTXTClustersCheckBox;
private javax.swing.JCheckBox contentTXTExpRatioCheckBox;
private javax.swing.JCheckBox contentTXTParametersUsedCheckBox;
private javax.swing.JCheckBox contentXLSClustersCheckBox;
private javax.swing.JCheckBox contentXLSExpRatioCheckBox;
private javax.swing.JLabel contentsLabel;
private javax.swing.JButton dataFileHelp;
private javax.swing.JLabel dataFileLabel;
private javax.swing.JTextField dataFilePath;
private javax.swing.JLabel destFolderLabel;
private javax.swing.JTextField destFolderPath;
private javax.swing.ButtonGroup distMetricGroup;
private javax.swing.JLabel distMetricLabel;
private javax.swing.JLabel errorMessageForm1;
private javax.swing.JLabel errorMessageForm2;
private javax.swing.JLabel errorMessageForm3;
private javax.swing.JLabel errorMessageForm4;
private javax.swing.JRadioButton euclidianRadio;
private javax.swing.JButton exportButton;
private javax.swing.JTextField exportExcelFileName;
private javax.swing.JTextField exportImageFileName;
private javax.swing.JTextField exportPDFFileName;
private javax.swing.JTextField exportTextFileName;
private javax.swing.JCheckBox exportToExcelCheckbox;
private javax.swing.JCheckBox exportToImageCheckbox;
private javax.swing.JCheckBox exportToPDFCheckbox;
private javax.swing.JCheckBox exportToTextCheckbox;
private javax.swing.JLabel fileFormatsLabel;
private javax.swing.JLabel fileNamesLabel;
private javax.swing.JLabel footnoteLabel1;
private javax.swing.JLabel footnoteLabel2;
private javax.swing.JPanel form1;
private javax.swing.JLabel form1Instruction;
private javax.swing.JPanel form2;
private javax.swing.JLabel form2Instruction;
private javax.swing.JPanel form3;
private javax.swing.JLabel form3Instruction;
private javax.swing.JPanel form4;
private javax.swing.JLabel geneDescriptionLabel;
private javax.swing.JButton geneFileHelp;
private javax.swing.JTextField geneFilePath;
private javax.swing.ButtonGroup genesOrSamplesgroup;
private javax.swing.JLabel initLearningRateLabel;
private javax.swing.JTextField initLearningRateText;
private javax.swing.JLayeredPane jLayeredPane1;
private javax.swing.JPanel jPanel3;
private javax.swing.JCheckBox logTransCheckbox;
private javax.swing.JButton logTransHelp;
private javax.swing.JPanel mainContentPanel;
private javax.swing.JButton mainHelpButton;
private javax.swing.JRadioButton manhattanRadio;
private javax.swing.JLabel maxIterLabel;
private javax.swing.JTextField maxIterText;
private javax.swing.JCheckBox medianCenteringCheckbox;
private javax.swing.JButton medianCenteringHelp;
private javax.swing.JPanel navigationButtonsPanel;
private javax.swing.JButton nextButton;
private javax.swing.JTextField numOfClustText;
private javax.swing.JLabel numberOfClustersLabel;
private javax.swing.JLabel pdfLabel;
private javax.swing.JLabel pngLabel;
private javax.swing.JButton preprocessButton;
private javax.swing.JPanel progressPanel;
private javax.swing.JProgressBar progressbar;
private javax.swing.JCheckBox quantileNormCheckbox;
private javax.swing.JButton quantileNormHelp;
private javax.swing.JButton resetButtonForm1;
private javax.swing.JButton resetButtonForm2;
private javax.swing.JButton resetButtonForm3;
private javax.swing.JButton resetButtonForm4;
private javax.swing.JLabel sampleDescriptionLabel;
private javax.swing.JButton sampleFileHelp;
private javax.swing.JTextField sampleFilePath;
private javax.swing.JCheckBox scaleNormCheckbox;
private javax.swing.JButton scaleNormHelp;
private javax.swing.JLabel status;
private javax.swing.JLabel stepBigImageLabel;
private javax.swing.JLabel stepDescription;
private javax.swing.JLabel stepTitle;
private javax.swing.JRadioButton supnormRadio;
private javax.swing.JLabel txtLabel;
private javax.swing.JButton uploadFileButton;
private javax.swing.JButton viewDataButton;
private javax.swing.JButton viewHeatMapButton;
private javax.swing.JLabel xlsLabel;
private javax.swing.JCheckBox zScoreTransCheckbox;
private javax.swing.JButton zScoreTransHelp;
private javax.swing.JFileChooser fileChooser = new JFileChooser();
File dataFile, geneFile, sampleFile;
private javax.swing.JFileChooser folderChooser = new JFileChooser();
uploadFileWorker uploadFile;
preprocessWorker preprocess;
clusterWorker cluster;
exportFileWorker exportFile;
FileParser fileParser;
Preprocess preprocessor;
SOM som;
KMeans kmeans;
ResultingData results;
FileExporter fileExporter;
public static double[][] data;
public static int[] clusterAssignments;
boolean isNewCluster, nextButtonEnabled, backButtonEnabled;
HeatMap heatmapPanel;
JFrame helpWindow = new HelpGUI();

private void resetData(){
        fileParser = null;
        preprocessor = null;
        som = null;
        kmeans = null;
        results = null;
        fileExporter = null;
        data = null;
        clusterAssignments = null;
        isNewCluster = true;
        heatmapPanel = null;
        System.gc();
}

class uploadFileWorker extends SwingWorker<String, String> {
        @Override
        public String doInBackground() {
                backButtonEnabled = backButton.isEnabled();
                nextButtonEnabled = nextButton.isEnabled();
                backButton.setEnabled(false);
        nextButton.setEnabled(false);
        fileParser = new FileParser();
        if(!dataFilePath.getText().equals("")) {
            if(this.isCancelled()) return "";
            status.setText("Parsing the data file...");
            progressbar.setValue(5);
            File file = new File(dataFilePath.getText());
            if(file.exists()) {
                fileParser.setDataFile(dataFilePath.getText());
                if(!fileParser.extractDataContents() ||
                !dataFilePath.getText().endsWith(".txt"))
                    return "The format for expression ratio is invalid.";
            }else return "File for expresion ratio was not found.";
            progressbar.setValue(35);
            if(!geneFilePath.getText().equals("")){
                if(this.isCancelled()) return "";
                status.setText("Parsing the gene descripton file...");
                progressbar.setValue(40);
                file = new File(geneFilePath.getText());
                if(file.exists()) {
                    fileParser.setGeneFile(geneFilePath.getText());
                    if(!fileParser. extractDescription
                    Contents(true)) // gene
                        return "The format for gene descriptions is
                        invalid.";
                }else return "File for gene description was not found.";
            }
            progressbar.setValue(70);
            if(!sampleFilePath.getText().equals("")){
                if(this.isCancelled()) return "";
                status.setText("Parsing the sample descriptions file...");
                progressbar.setValue(75);
                file = new File(sampleFilePath.getText());
                if(file.exists()) {
                    fileParser.setSampleFile(
                    sampleFilePath.getText());
                    if(!fileParser.extractDescription
                    Contents(false))//samples
                        return "The format for sample descriptions is
                        invalid." ;
                }else
                    return "File for sample description was not found.";
            }
        }else return "The data file is required.";
        return "";
    }

    @Override
    public void done() {
        Toolkit.getDefaultToolkit().beep();
        try {
            errorMessageForm1.setText(get());
        } catch(CancellationException e){
            status.setText("Data submission is cancelled.");
            progressbar.setValue(0);
        } catch (InterruptedException  | ExecutionException e) {
            errorMessageForm1.setText("Problem occured.");
        }
        if(!this.isCancelled()){
            if(errorMessageForm1.getText().equals("")){
                nextButtonEnabled = true;
                status.setText("Done submitting.");
                progressbar.setValue(100);
            } else errorMessageForm1.setVisible(true);
        }
        nextButton.setEnabled(nextButtonEnabled);
        backButton.setEnabled(backButtonEnabled);
        backgroundPanel.setCursor(Cursor.getPredefinedCursor
        (Cursor.DEFAULT_CURSOR));
    }
}
class preprocessWorker extends SwingWorker<Boolean, Void> {
    @Override
    public Boolean doInBackground() {
        backButtonEnabled = backButton.isEnabled();
        nextButtonEnabled = nextButton.isEnabled();
        backButton.setEnabled(false);
        nextButton.setEnabled(false);
        boolean isPreprocessed = false;
        preprocessor = null;
        preprocessor = new Preprocess(fileParser.data);
        if( logTransCheckbox.isSelected()){
            if(this.isCancelled()) return false;
            status.setText("Performing logarithmic transformation...");
            progressbar.setValue(5);
            preprocessor.logTransform();
            progressbar.setValue(20);
            isPreprocessed = true;
        }
        if( zScoreTransCheckbox.isSelected()){
            if(this.isCancelled()) return false;
            status.setText("Performing Z-Score transformation...");
            progressbar.setValue(25);
            preprocessor.zScoreTransform();
            progressbar.setValue(40);
            isPreprocessed = true;
        }
        if( medianCenteringCheckbox.isSelected()){
            if(this.isCancelled()) return false;
            status.setText("Performing median centering...");
            progressbar.setValue(45);
            preprocessor.medianCenter();
            progressbar.setValue(50);
            isPreprocessed = true;
        }
        if( scaleNormCheckbox.isSelected()){
            if(this.isCancelled()) return false;
            status.setText("Performing scale normalization...");
            progressbar.setValue(55);
            preprocessor.scaleNormalize();
            progressbar.setValue(80);
            isPreprocessed = true;
        }
        if( quantileNormCheckbox.isSelected()){
            if(this.isCancelled()) return false;
            status.setText("Performing quantile normalization...");
            progressbar.setValue(85);
            preprocessor.quantileNormalize();
            isPreprocessed = true;
        }
        preprocessor.rawData = null;
        System.gc();
        return isPreprocessed;
    }
    @Override
    public void done() {
        Toolkit.getDefaultToolkit().beep();
        try {
            if(get()){
                nextButton.setText("Next");
                status.setText("Done preprocessing.");
                progressbar.setValue(100);
            }else{
                errorMessageForm2.setText("No preprocessing method chosen.
                You may skip this step if unnnecessary.");
                errorMessageForm2.setVisible(true);
                status.setText("\t");
                progressbar.setValue(0);
            }
        } catch(CancellationException e){
            status.setText("Preprocessing is cancelled.");
            progressbar.setValue(0);
            preprocessor = null; /// reset to raw data
        } catch (InterruptedException|ExecutionException e) {
            errorMessageForm2.setText("Problem occured.");
        }
        nextButton.setEnabled(nextButtonEnabled);
        backButton.setEnabled(backButtonEnabled);
        backgroundPanel.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSO
        R));
    }
}

class clusterWorker extends SwingWorker<String, Void> {
    @Override
    public String doInBackground() {
        backButtonEnabled = backButton.isEnabled();
        nextButtonEnabled = nextButton.isEnabled();
        backButton.setEnabled(false);
        nextButton.setEnabled(false);
        isNewCluster = true;
        int clusters=0, maxIter=0;
        double initLearningRate=0;
        try{
            clusters = Integer.parseInt(numOfClustText.getText());
            if(clusters<=0) return "Number of clusters must be greater than
        zero.";
        }catch(NumberFormatException e){ return "Invalid number of clusters."; }
        try{
            initLearningRate =
            Double.parseDouble(initLearningRateText.getText());
            if(initLearningRate<0 || initLearningRate>1) return "Initial
            learning rate must be between 0 and 1.";
        }catch(NumberFormatException e){ return "Invalid initial learning
        rate."; }
        try{
            maxIter = Integer.parseInt(maxIterText.getText());
            if(maxIter<0) return "Number of iterations must be greater than
            zero.";
        }catch(NumberFormatException e){ return "Invalid number of iterations.";
        }

        int distMetric = 1;
        if(euclidianRadio.isSelected()) distMetric = 2;
        else if (supnormRadio.isSelected()) distMetric = 3;
        boolean isClusterSamples = clusterBySamplesRadio.isSelected();
        if(this.isCancelled()) return "";
        status.setText("Training self-organizing map...");
        progressbar.setValue(5);
```

```
                som = null;
                kmeans = null;
                results = null;
                fileExporter = null;
                data = null;
                clusterAssignments = null;
                heatmapPanel = null;
                System.gc();
                if(preprocessor==null)
                        som = new SOM(fileParser.data, clusters, maxIter, distMetric,
                        initLearningRate,isClusterSamples);
                else som = new SOM(preprocessor.data, clusters, maxIter, distMetric,
                initLearningRate,isClusterSamples);
                som.train();
                if(this.isCancelled()) return "";
                status.setText("Clustering...");
                progressbar.setValue(70);
                kmeans=new KMeans(som.getCentroids(), som.data, som.distMetric);
                clusterAssignments = kmeans.groupData();
                data = kmeans.arrangeDataIntoGroups();
                if(isClusterSamples) data = som.invertArray(data,data[0].length);
                return "";
        }

        @Override
        public void done() {
                Toolkit.getDefaultToolkit().beep();
                try {
                        errorMessageForm3.setText(get());
                } catch(CancellationException e){
                        status.setText("Clustering is cancelled.");
                        progressbar.setValue(0);
                }catch (InterruptedException | ExecutionException e) {
                        errorMessageForm3.setText("Problem occured.");
                }
                if(!this.isCancelled()){
                        if(errorMessageForm3.getText().equals("")){
                                nextButtonEnabled = true;
                                status.setText("Done clustering.");
                                progressbar.setValue(100);
                        } else errorMessageForm3.setVisible(true);
                }
                nextButton.setEnabled(nextButtonEnabled);
                backButton.setEnabled(backButtonEnabled);
                backgroundPanel.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSO
                R));
        }
}

public void displayDataResults(JPanel heatmap){
        results = new ResultingData(heatmap,kmeans.data);
        if(preprocessor==null)
                results.getPreprocMethodsUsed(false,false,false,false,false);
        else results.getPreprocMethodsUsed(preprocessor.isLogTransformed,
        preprocessor.isMedianCentered,
        preprocessor.isQuantileNormalized,preprocessor.isScaleNormalized,preprocessor.i
        sZScoreTransformed);
                results.getClusteringParameters(som.clusters, som.isByCols,
        som.initLearningRate, som.maxIter, som.distMetric);
        results.getClusters(kmeans.clusters,kmeans.numberofResultingClusters);
                results.getClusterAssignments(kmeans.clusterAssignment);
                results.getClusterVariances();
                results.getOverAllMSD();
```

```
	results.getOriginalData(fileParser.data,fileParser.dataRowLabels,fileParser.dat
	aColLabels);
	results.getDataDescriptions(fileParser.dataRowDescription,fileParser.dataColDes
	cription);
	results.generateParamUsed();
	results.generateClustProperties();
	results.generateExpRatio();
	results.showWindow();
}


class exportFileWorker extends SwingWorker<String, Void> {
	@Override
	public String doInBackground() {
		backButtonEnabled = backButton.isEnabled();
		nextButtonEnabled = nextButton.isEnabled();
		backButton.setEnabled(false);
		nextButton.setEnabled(false);
		fileExporter = new FileExporter();
		if(!(exportToImageCheckbox.isSelected() ||
		exportToExcelCheckbox.isSelected() || exportToPDFCheckbox.isSelected() ||
		exportToTextCheckbox.isSelected()))
			return "Please choose at least one file to export.";
		else if(destFolderPath.getText().equals(""))
			return "Select where to put the files.";
		else if
		(!fileExporter.canWriteToDirectory(destFolderPath.getText()+"\\temp.txt")
		) return "Invalid destination folder.";
		else{
			if(exportToImageCheckbox.isSelected()){ //image
				if(exportImageFileName.getText().equals(""))
					return "File name for the image is required.";
				else if(!fileExporter.isFilenameValid
				(exportImageFileName.getText()+".png"))
					return "File name for the image is invalid.";
				else{ // generate image
					if(this.isCancelled()) return "";
					status.setText("Exporting image file...");
					progressbar.setValue(5);
					if(!fileExporter.saveHeatmapPanelToImage(heatmapPanel
					,destFolderPath.getText()+"\\"+exportImageFileName.ge
					tText()+".png"))
						return "Problem occured while exporting the
						image";
					progressbar.setValue(25);
				}
			}
			if(exportToExcelCheckbox.isSelected()){//excel
				if(exportExcelFileName.getText().equals(""))
					return "File name for the spreadsheet is required.";
				else if(!fileExporter.isFilenameValid(
				exportExcelFileName.getText()+".xls"))
					return "File name for the spreadsheet is invalid.";
				else if(contentXLSClustersCheckBox.isSelected() ||
				contentXLSExpRatioCheckBox.isSelected()){
					if(this.isCancelled()) return "";
					status.setText("Exporting spreadsheet...");
					progressbar.setValue(30);
					ArrayList<JTable> tables = new ArrayList<>();
					// generate excel: clusters or exp ratio
					if(contentXLSClustersCheckBox.isSelected()){
						JTable temp = new JTable(new
					DefaultTableModel(new Object[][]{{"Number of
                clusters", results.numOfFinalClusters},{"Mean Squared
                Distances of Centroids", results.overAllVariance}
                },new Object[]{"",""} ));
                tables.add(temp);
                tables.add(results.rgui.clustPropertiesTableWithOutBu
                ttons);
                }
                if(contentXLSExpRatioCheckBox.isSelected())
                        tables.add(results.rgui.expRatioTable);
                if(!fileExporter.exportTablesToFile(tables,
                destFolderPath.getText()+"\\"+exportExcelFileName.get
                Text()+".xls"))
                        return "Problem occured while exporting the
                        spreadsheet.";
                progressbar.setValue(50);
        }else return "Choose what contents to include in
                the spreadsheet.";
}
if(exportToTextCheckbox.isSelected()){//text
        if(exportTextFileName.getText().equals(""))
        return "File name for the text file is required.";
        else if(!fileExporter.isFilenameValid(
        exportTextFileName.getText()+".txt"))
                return "File name for the text file is invalid.";
        else if( ontentTXTParametersUsedCheckBox.isSelected() ||
        contentTXTExpRatioCheckBox.isSelected() ||
        contentTXTClustersCheckBox.isSelected()){
                if(this.isCancelled()) return "";
                status.setText("Exporting text file...");
                progressbar.setValue(55);
                ArrayList<JTable> tables = new ArrayList<>();
                // generate text file: params or clusters or exp
                ratio
        if(contentTXTParametersUsedCheckBox.isSelected())
                tables.add(results.rgui.paramUsedTable);
                if(contentTXTClustersCheckBox.isSelected()){
                        JTable temp = new JTable(new
                        DefaultTableModel(new Object[][]{
                        {"Number of clusters", results.
                        numOfFinalClusters},{"Mean Squared Distances
                        of Centroids", results.overAllVariance} },new
                        Object[]{"",""} ));
                        tables.add(temp);
                        tables.add(results.rgui.clustPropertiesTableWi
                        thOutButtons);
                }
                if(contentTXTExpRatioCheckBox.isSelected())
                        tables.add(results.rgui.expRatioTable);
                if(!fileExporter.exportTablesToFile(tables,
                destFolderPath.getText()+"\\"+exportTextFileName.getT
                ext()+".txt"))
                        return "Problem occured while exporting the
                        text file.";
                progressbar.setValue(75);
        }else return "Choose what contents to include in the
                text file.";
}
if(exportToPDFCheckbox.isSelected()){//pdf
        if(exportPDFFileName.getText().equals(""))
                return "File name for the PDF file is required.";
        else if(!fileExporter.isFilenameValid(
        exportPDFFileName.getText()+".pdf"))
                return "File name for the PDF file is invalid.";
                else if(contentPDFParametersUsedCheckBox.isSelected() ||
                contentPDFExpRatioCheckBox.isSelected() ||
                contentPDFClustersCheckBox.isSelected() ||
                contentPDFHeatMapCheckBox.isSelected() ){
                    if(this.isCancelled()) return "";
                        status.setText("Exporting PDF file...");
                    progressbar.setValue(80);
                    ArrayList<JTable> tables = new ArrayList<>();
                    // generate PDF file: params or heatmap or
                    clusters or exp ratio
                    if(contentPDFParametersUsedCheckBox.
                    isSelected())
                        tables.add(results.rgui.paramUsedTable);
                    if(contentPDFClustersCheckBox.isSelected()){
                        JTable temp = new JTable(new
                        DefaultTableModel(new Object[][]{{"Number of
                        clusters", results.numOfFinalClusters},{"Mean
                        Squared Distances of Centroids",
                        results.overAllVariance} },new
                        Object[]{"Results",""} ));
                        tables.add(temp);
                        tables.add(results.rgui.clustPropertiesTableWi
                        thOutButtons);
                    }
                    if(contentPDFExpRatioCheckBox.isSelected())
                        tables.add(results.rgui.expRatioTable);
                    JPanel temp = null;
                    if(contentPDFHeatMapCheckBox.isSelected())
                        temp = heatmapPanel;
                    if(!fileExporter.exportToPDF(tables,
                    destFolderPath.getText()+"\\"+exportPDFFileName.getTe
                    xt()+".pdf", temp)) return "Problem occured while
                    exporting the PDF file.";
                }else return "Choose what contents to include in the
                    PDF file.";
            }
        }
        return "";
    }

        @Override
        public void done() {
            Toolkit.getDefaultToolkit().beep();
            try {
                errorMessageForm4.setText(get());
            } catch(CancellationException e){
                status.setText("File exporting is cancelled.");
                progressbar.setValue(0);
            } catch (InterruptedException | ExecutionException e) {
                errorMessageForm4.setText("Problem occured.");
            }
            if(errorMessageForm4.getText().equals("") && !this.isCancelled()){
                status.setText("Done exporting.");
                progressbar.setValue(100);
            }
            nextButton.setEnabled(nextButtonEnabled);
            backButton.setEnabled(backButtonEnabled);
            backgroundPanel.setCursor(Cursor.getPredefinedCursor(
            Cursor.DEFAULT_CURSOR));
        }
    }
}
```

M. MaSOM/src/HelpGUI.java

```
import java.net.URL;
import javax.swing.ImageIcon;

@SuppressWarnings("serial")
public class HelpGUI extends javax.swing.JFrame {
       public HelpGUI(int p) {
        this();
       for(int i = 0; i< 9; i++)page[i].setVisible(false);
       page[p-1].setVisible(true);
       next.setEnabled(true);
       back.setEnabled(true);
       this.setVisible(true);
    }
    public HelpGUI() {
        initComponents();
        page[0].setVisible(true);
        for(int i=1; i<9; i++) page[i].setVisible(false);
        back.setEnabled(false);
        this.setDefaultCloseOperation(HIDE_ON_CLOSE);
    }
    private void initComponents() {
       page = new javax.swing.JPanel[9];
        background = new javax.swing.JPanel();
        header = new javax.swing.JPanel();
        jLabel45 = new javax.swing.JLabel();
        footer = new javax.swing.JPanel();
        back = new javax.swing.JButton();
        next = new javax.swing.JButton();
        body = new javax.swing.JLayeredPane();
        page[8] = new javax.swing.JPanel();
        jLabel50 = new javax.swing.JLabel();
        jLabel51 = new javax.swing.JLabel();
        jLabel52 = new javax.swing.JLabel();
        jLabel53 = new javax.swing.JLabel();
        page[0] = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        page[1] = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        page[2] = new javax.swing.JPanel();
        jLabel10 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        page[3] = new javax.swing.JPanel();
        jLabel9 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        jLabel16 = new javax.swing.JLabel();
        page[4] = new javax.swing.JPanel();
        jLabel17 = new javax.swing.JLabel();
        jLabel18 = new javax.swing.JLabel();
        jLabel19 = new javax.swing.JLabel();
        jLabel20 = new javax.swing.JLabel();
        jLabel21 = new javax.swing.JLabel();
        jLabel22 = new javax.swing.JLabel();
        jLabel23 = new javax.swing.JLabel();
        jLabel24 = new javax.swing.JLabel();
        page[5] = new javax.swing.JPanel();
        jLabel25 = new javax.swing.JLabel();
        jLabel26 = new javax.swing.JLabel();
        jLabel27 = new javax.swing.JLabel();
        jLabel28 = new javax.swing.JLabel();
        jLabel29 = new javax.swing.JLabel();
        jLabel30 = new javax.swing.JLabel();
        page[6] = new javax.swing.JPanel();
        jLabel31 = new javax.swing.JLabel();
        jLabel32 = new javax.swing.JLabel();
        jLabel33 = new javax.swing.JLabel();
        jLabel34 = new javax.swing.JLabel();
        jLabel37 = new javax.swing.JLabel();
        jLabel38 = new javax.swing.JLabel();
        page[7] = new javax.swing.JPanel();
        jLabel35 = new javax.swing.JLabel();
        jLabel36 = new javax.swing.JLabel();
        jLabel39 = new javax.swing.JLabel();
        jLabel40 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("User Guide");
        setBackground(new java.awt.Color(255, 255, 255));
        setResizable(false);

       URL imgUrl = getClass().getResource("src/img/main.png");
       if(imgUrl !=null)
             super.setIconImage(new ImageIcon(imgUrl).getImage());
        else super.setIconImage(new ImageIcon("src/img/main.png").getImage());
        background.setBackground(new java.awt.Color(255, 255, 255));
       header.setOpaque(false);
        imgUrl = getClass().getResource("src/img/help_banner.png");
             if(imgUrl !=null)
                    jLabel45.setIcon(new ImageIcon(imgUrl));
             else jLabel45.setIcon(new
             javax.swing.ImageIcon("src/img/help_banner.png"));

         javax.swing.GroupLayout headerLayout = new javax.swing.GroupLayout(header);
        header.setLayout(headerLayout);
        headerLayout.setHorizontalGroup(

headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jLabel45, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE));
headerLayout.setVerticalGroup(headerLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING).addComponent(jLabel45, javax.swing.GroupLayout.DEFAULT_SIZE, 31,
Short.MAX_VALUE));
        footer.setOpaque(false);
        imgUrl = getClass().getResource("src/img/help_back.png");
         if(imgUrl !=null) back.setIcon(new ImageIcon(imgUrl));
         else  back.setIcon(new javax.swing.ImageIcon ("src/img/help_back.png"));
        back.setContentAreaFilled(false);
        back.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                backActionPerformed(evt);
            }
        });
        imgUrl = getClass().getResource("src/img/help_next.png");
         if(imgUrl !=null)next.setIcon(new ImageIcon(imgUrl));
         else next.setIcon(new javax.swing.ImageIcon      ("src/img/help_next.png"));
        next.setContentAreaFilled(false);
        next.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                nextActionPerformed(evt);
            }
        });
         javax.swing.GroupLayout footerLayout = new javax.swing.GroupLayout(footer);
        footer.setLayout(footerLayout);
        footerLayout.setHorizontalGroup(
footerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(footerLayout.createSequentialGroup().addGap(122, 122, 122)
.addComponent(back).addGap(91, 91, 91).addComponent(next) .addContainerGap(119,
Short.MAX_VALUE)));
        footerLayout.setVerticalGroup(
footerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addGroup(footerLayout.createSequentialGroup()
.addGroup(footerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(back).addComponent(next)).addGap(0, 22, Short.MAX_VALUE)));
        body.setBackground(new java.awt.Color(255, 255, 255));
        body.setOpaque(true);
        page[8].setBackground(new java.awt.Color(255, 255, 255));
        jLabel50.setBackground(new java.awt.Color(220, 220, 230));
        jLabel50.setText("   B.  Summary");
        jLabel50.setOpaque(true);
        jLabel51.setFont(new java.awt.Font("Verdana", 0, 10));
jLabel51.setText("<html><p align=justify>Three tables are available to analyze the
results of the clustering. First is the Parameters Used table which displays the
inputs used to cluster the data.  Another is the Cluster Properties which summarizes
the characteristics of each cluster. In this table, the user may export the data or
the description of a specific cluster which can be further clustered when hierarchical
clustering is desired. Expression Ratios table lists all the genes or samples that are
clustered together with the supplementary information for each, if available.");
        jLabel52.setBackground(new java.awt.Color(230, 220, 230));
        jLabel52.setText("   C.  Export");
        jLabel52.setOpaque(true);
        jLabel53.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel53.setText("<html><p align=justify>The tool provides export facilities
for archiving and documentations of the clustering activities. Results can be exported
into several files depending on what content and format are needed by the user.");
         javax.swing.GroupLayout page9Layout = new javax.swing.GroupLayout(page[8]);
        page[8].setLayout(page9Layout);
        page9Layout.setHorizontalGroup(
            page9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page9Layout.createSequentialGroup().addGap(0, 0, Short.MAX_VALUE)

.addGroup(page9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false).addComponent(jLabel50, javax.swing.GroupLayout.DEFAULT_SIZE, 382,
Short.MAX_VALUE).addComponent(jLabel52, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page9Layout.createSequentialGroup().addContainerGap(62, Short.MAX_VALUE)

.addGroup(page9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page9Layout.createSequentialGroup().addComponent(jLabel51,
javax.swing.GroupLayout.PREFERRED_SIZE, 329,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(29, 29, 29))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page9Layout.createSequentialGroup().addComponent(jLabel53,
javax.swing.GroupLayout.PREFERRED_SIZE, 330,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(26, 26, 26)))) );
page9Layout.setVerticalGroup(page9Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(page9Layout.createSequentialGroup().addGap(59, 59,
59).addComponent(jLabel50, javax.swing.GroupLayout.PREFERRED_SIZE, 27,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18) .addComponent(jLabel51,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(33, 33, 33).addComponent(jLabel52, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18) .addComponent(jLabel53,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(51, Short.MAX_VALUE)));

         page[8].setBounds(0, 10, 420, 430);
        body.add(page[8], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[0].setBackground(new java.awt.Color(255, 255, 255));
        jLabel1.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel1.setText("<html><p align=justify>Welcome to MaSOM - a tool intended to
help researchers in analyzing image-processed two-color cDNA microarray data. This
program uses Self-Organizing Maps to cluster expression ratios such that data of the
same cluster are relatively similar to each other while those belonging to different
groups are dissimilar.");
        jLabel2.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel2.setText("<html><p align=justify>This program takes four main steps to
cluster microarray data: Data Input, Data Preprocessing, Data Clustering, and Results
Visualization and Summary.");
        jLabel3.setBackground(new java.awt.Color(220, 240, 250));
        jLabel3.setFont(new java.awt.Font("Tahoma", java.awt.Font.BOLD, 14));
        jLabel3.setText("        MaSOM");
        jLabel3.setOpaque(true);
        jLabel4.setBackground(new java.awt.Color(220, 220, 230));
        jLabel4.setFont(new java.awt.Font("Verdana",
         java.awt.Font.BOLD, 10));
        jLabel4.setText("    Microarray Self-Organizing Maps");
        jLabel4.setOpaque(true);
         javax.swing.GroupLayout page1Layout = new
         javax.swing.GroupLayout(page[0]);
        page[0].setLayout(page1Layout);
        page1Layout.setHorizontalGroup(
            page1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(page1Layout.createSequentialGroup().addContainerGap(javax.swing.GroupLayout.
DEFAULT_SIZE, Short.MAX_VALUE)
.addGroup(page1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 395,
javax.swing.GroupLayout.PREFERRED_SIZE).addGroup(page1Layout.createSequentialGroup().a
ddGap(18, 18, 18)
.addGroup(page1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false).addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE).addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 335,
Short.MAX_VALUE))))
.addContainerGap()).addGroup(page1Layout.createSequentialGroup()
.addGap(18, 18, 18).addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 410,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap(javax.swing.GroupLayout.DEFAUL
T_SIZE, Short.MAX_VALUE)));
page1Layout.setVerticalGroup(page1Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(page1Layout.createSequentialGroup()
.addGap(66, 66, 66).addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.UNRELATED).addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE,
26, javax.swing.GroupLayout.PREFERRED_SIZE).addGap(28, 28, 28)
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18) .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0,105,Short.MAX_VALUE)));

        page[0].setBounds(0, 10, 420, 430);
        body.add(page[0], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[1].setBackground(new java.awt.Color(255, 255, 255));
        jLabel5.setFont(new java.awt.Font("Verdana", 0, 10));
jLabel5.setText("<html><p align=justify>In this phase, the user is expected to provide
a file containing the microarray data to be clustered. Supplementary data for the
genes and samples may also be supplied for further analysis.");
        jLabel6.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel6.setText("<html><p align=justify>This text file must contain the
numerical expression ratios of each probe's redness over greenness. Its content is
basically a tab-delimited table of values wherein the first row is the list of the
identifiers or labels for their respective columns and the values at the first column
are the identifiers for their respective rows. Each of the rows, except the first,
corresponds to the ratio of a particular gene in varying samples while each of the
columns, except the first, pertains to a specific sample's ratio for every gene.");
        jLabel7.setBackground(new java.awt.Color(220, 230, 250));
        jLabel7.setText("   I.  Data Input ");
        jLabel7.setOpaque(true);
        jLabel8.setBackground(new java.awt.Color(220, 220, 230));
        jLabel8.setText("   A.  Data file ");
        jLabel8.setOpaque(true);
         javax.swing.GroupLayout page2Layout = new  javax.swing.GroupLayout(page[1]);
        page[1].setLayout(page2Layout);
        page2Layout.setHorizontalGroup(
            page2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(page2Layout.createSequentialGroup().addGap(25, 25, 25)
.addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 403,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap())
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page2Layout.createSequentialGroup().addContainerGap(javax.swing.GroupLayout.DEFAULT_SI
ZE, Short.MAX_VALUE)
.addGroup(page2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page2Layout.createSequentialGroup().addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 378,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap())
.addGroup(page2Layout.createSequentialGroup().addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 335,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap()))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page2Layout.createSequentialGroup().addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 304,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(55, 55, 55)))));

page2Layout.setVerticalGroup(page2Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(page2Layout.createSequentialGroup()
.addGap(34, 34, 34).addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18).addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(26, 26, 26).addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18).addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 165,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0, 44,Short.MAX_VALUE)));

        page[1].setBounds(0, 10, 420, 430);
        body.add(page[1], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[2].setBackground(new java.awt.Color(255, 255, 255));
        jLabel10.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel10.setText("<html><p align=justify>This optional text file contains a
tab-delimited table where the first row is the list of the samples' properties and the
values at the first column are the identifiers for the samples of the data file. Each
of the rows, except the first, is the corresponding sample’s characteristic at
respective properties.");
        jLabel12.setBackground(new java.awt.Color(220, 220, 230));
        jLabel12.setText("   B.  Gene description file ");
        jLabel12.setOpaque(true);
        jLabel13.setBackground(new java.awt.Color(230, 220, 230));
        jLabel13.setText("   C.  Sample description file ");
        jLabel13.setOpaque(true);
        jLabel11.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel11.setText("<html><p align=justify>This optional text file contains a
tab-delimited table where the first row is the list of the genes’ properties and the
values at the first column are the identifiers for the genes of the data file. Each of
the rows, except the first, is the corresponding gene’s characteristic at respective
properties.");

         javax.swing.GroupLayout page3Layout = new javax.swing.GroupLayout(page[2]);
        page[2].setLayout(page3Layout);
        page3Layout.setHorizontalGroup(
            page3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(page3Layout.createSequentialGroup().addGap(34, 34, 34)

.addGroup(page3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(page3Layout.createSequentialGroup().addComponent(jLabel12,
javax.swing.GroupLayout.PREFERRED_SIZE, 392,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0, 0, Short.MAX_VALUE))
.addComponent(jLabel13, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)).addContainerGap())
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page3Layout.createSequentialGroup().addContainerGap(javax.swing.GroupLayout.DEFAULT_SI
ZE, Short.MAX_VALUE)
.addGroup(page3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page3Layout.createSequentialGroup().addComponent(jLabel10,
javax.swing.GroupLayout.PREFERRED_SIZE, 323,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(48, 48, 48))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page3Layout.createSequentialGroup().addComponent(jLabel11,
javax.swing.GroupLayout.PREFERRED_SIZE, 328,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(43, 43, 43)))));
        page3Layout.setVerticalGroup(
page3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(page3Layout.createSequentialGroup().addGap(56, 56, 56)
.addComponent(jLabel12, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18).addComponent(jLabel11,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(28, 28, 28)
.addComponent(jLabel13, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18)
.addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED_SIZE, 113,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0,48,Short.MAX_VALUE)));

        page[2].setBounds(0, 10, 420, 430);
        body.add(page[2], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[3].setBackground(new java.awt.Color(255, 255, 255));
        jLabel9.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel9.setText("<html><p align=justify>In this step, the user may perform
initial adjustments to the data in order to remove experimental bias and other
possible non-biological variations that were accumulated from the microarray
preprocessing to its image processing. The tool offers the following preprocessing
techniques that may also be performed jointly. Note that the tool will only perform
the chosen process and that, whenever the user re-preprocesses the data, it resets the
data to its original values before preprocessing. This is to make sure that each
method will only be applied at most once.");
```

```
        jLabel14.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel14.setText("<html><p align=justify>This transformation uses binary
logarithm formula to adjust the distribution of the values into a logarithmic curve.
It transforms the large values, or very red spots, into smaller positive numbers and
the small values, or very green spots, into a negative numbers. Values equal to one,
which corresponds to spots that are yellow, are converted into zeros. In this manner,
it is assured that vectors of data have negative values for down-regulated genes, zero
for equally-expressed genes, and positive values for up-regulated genes.");
        jLabel15.setBackground(new java.awt.Color(220, 230, 250));
        jLabel15.setText("   II.  Data Preprocessing ");
        jLabel15.setOpaque(true);
        jLabel16.setBackground(new java.awt.Color(220, 220, 230));
        jLabel16.setText("   A.  Logarithmic Transformation ");
        jLabel16.setOpaque(true);

         javax.swing.GroupLayout page4Layout = new javax.swing.GroupLayout(page[3]);
        page[3].setLayout(page4Layout);
        page4Layout.setHorizontalGroup(
            page4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(page4Layout.createSequentialGroup().addGroup(page4Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING).addGroup(page4Layout.createSequentialGroup
().addGap(24, 24, 24).addComponent(jLabel15, javax.swing.GroupLayout.PREFERRED_SIZE,
404,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGroup(page4Layout.createSequentialGroup().
addGap(47, 47, 47)
.addGroup(page4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddComponent(jLabel16, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addGroup(page4Layout.createSequentialGroup() .addComponent(jLabel9,
javax.swing.GroupLayout.PREFERRED_SIZE, 345,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0, 0, Short.MAX_VALUE)))))
.addContainerGap()).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page4Layout.createSequentialGroup().addGap(0, 0, Short.MAX_VALUE)
.addComponent(jLabel14, javax.swing.GroupLayout.PREFERRED_SIZE, 315,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(50, 50, 50)));

page4Layout.setVerticalGroup(page4Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(page4Layout.createSequentialGroup().addGap(34, 34,
34).addComponent(jLabel15, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(29, 29, 29)
.addComponent(jLabel16, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18).addComponent(jLabel14,
javax.swing.GroupLayout.PREFERRED_SIZE, 135,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0,7,Short.MAX_VALUE)));

        page[3].setBounds(0, 10, 420, 430);
        body.add(page[3], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[4].setBackground(new java.awt.Color(255, 255, 255));
        jLabel17.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel17.setText("<html><p align=justify>In this method, the data points are
subtracted by the median of their vector to convert the points on the lower half into
negative numbers and points on the upper half into positive numbers. All vectors will
then have medians equal to zero.");
        jLabel18.setBackground(new java.awt.Color(210, 220, 230));
        jLabel18.setText("   B.  Z-Score Transformation ");
        jLabel18.setOpaque(true);
        jLabel19.setBackground(new java.awt.Color(220, 220, 230));
        jLabel19.setText("   C.  Median Centering ");
        jLabel19.setOpaque(true);
        jLabel20.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel20.setText("<html><p align=justify>This technique uses the Z-Score
formula in order to assure that all the data vectors have mean of 0 and variance of
1.");
        jLabel21.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel21.setText("<html><p align=justify>This technique forces the vectors to
have the same quantiles or empirical distributions by averaging the data points that
belong to the same ranks and then replaces those points with the computed mean. ");
        jLabel22.setBackground(new java.awt.Color(240, 220, 230));
        jLabel22.setText("   E.  Quantile Normalization ");
        jLabel22.setOpaque(true);
        jLabel23.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel23.setText("<html><p align=justify>This process adjusts the mean
centered log-transformed ratios so that all groups have the same variances and
means.");
        jLabel24.setBackground(new java.awt.Color(230, 220, 230));
        jLabel24.setText("   D.  Scale Normalization ");
        jLabel24.setOpaque(true);

         javax.swing.GroupLayout page5Layout = new javax.swing.GroupLayout(page[4]);
        page[4].setLayout(page5Layout);

page5Layout.setHorizontalGroup(page5Layout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page5Layout.createSequentialGroup().addGap(0, 0, Short.MAX_VALUE)
.addGroup(page5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false).addComponent(jLabel18, javax.swing.GroupLayout.DEFAULT_SIZE, 390,
Short.MAX_VALUE).addComponent(jLabel19, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE).addComponent(jLabel24,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE).addComponent(jLabel22,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page5Layout.createSequentialGroup().addContainerGap(54, Short.MAX_VALUE)
.addGroup(page5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page5Layout.createSequentialGroup().addComponent(jLabel20,
javax.swing.GroupLayout.PREFERRED_SIZE, 342,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(24, 24,
24)).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page5Layout.createSequentialGroup().addGroup(page5Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING).addComponent(jLabel17,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
338, javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(jLabel23,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
337, javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(27, 27, 27))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page5Layout.createSequentialGroup().addComponent(jLabel21,
javax.swing.GroupLayout.PREFERRED_SIZE, 340,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(25, 25, 25))))
); page5Layout.setVerticalGroup(
page5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(page5Layout.createSequentialGroup().addContainerGap()
.addComponent(jLabel18, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.RELATED).addComponent(jLabel20, javax.swing.GroupLayout.PREFERRED_SIZE,
43,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.UNRELATED).addComponent(jLabel19, javax.swing.GroupLayout.PREFERRED_SIZE,
26, javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jLabel17, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.RELATED, 18, Short.MAX_VALUE).addComponent(jLabel24,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.UNRELATED).addComponent(jLabel23, javax.swing.GroupLayout.PREFERRED_SIZE,
43, javax.swing.GroupLayout.PREFERRED_SIZE).addGap(20, 20, 20) .addComponent(jLabel22,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.UNRELATED).addComponent(jLabel21, javax.swing.GroupLayout.PREFERRED_SIZE,
65, javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap()));

        page[4].setBounds(0, 10, 420, 430);
        body.add(page[4], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[5].setBackground(new java.awt.Color(255, 255, 255));
        jLabel25.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel25.setText("<html><p align=justify>This phase is when the actual
grouping of the microarray data takes place. In this tool, microarray data is being
clustered by means of Self-Organizing Maps and K-Means. This method, in order to
discover the natural topology of the data, requires several parameters which
influences its manner of processing and, consequently, its results.");
        jLabel26.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel26.setText("<html><p align=justify>The user must provide the desired
maximum possible number of clusters to result from the process as this number is to be
used by the neural network as its map dimension. Take note that it is likely for the
Self-Organizing Map to output clusters less than what was provided by the user.");
        jLabel27.setBackground(new java.awt.Color(220, 230, 250));
        jLabel27.setText("   III.  Data Clustering ");
        jLabel27.setOpaque(true);
        jLabel28.setBackground(new java.awt.Color(220, 220, 230));
        jLabel28.setText("   A.  Number of clusters ");
        jLabel28.setOpaque(true);
        jLabel29.setBackground(new java.awt.Color(230, 220, 230));
        jLabel29.setText("   B.  Clustering by Samples or by Genes ");
        jLabel29.setOpaque(true);
        jLabel30.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel30.setText("<html><p align=justify>Clustering may be done either onto
samples or onto genes depending on the user’s need.");
         javax.swing.GroupLayout page6Layout = new javax.swing.GroupLayout(page[5]);
        page[5].setLayout(page6Layout);

page6Layout.setHorizontalGroup(page6Layout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING).addGroup(page6Layout.createSequentialGroup()
.addGroup(page6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(page6Layout.createSequentialGroup().addGap(25, 25, 25)
.addComponent(jLabel27, javax.swing.GroupLayout.PREFERRED_SIZE, 403,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(0, 0, Short.MAX_VALUE))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page6Layout.createSequentialGroup().addGap(0, 0, Short.MAX_VALUE)
.addGroup(page6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddComponent(jLabel25, javax.swing.GroupLayout.PREFERRED_SIZE, 340,
javax.swing.GroupLayout.PREFERRED_SIZE).addGroup(page6Layout.createParallelGroup(javax
.swing.GroupLayout.Alignment.LEADING, false).addComponent(jLabel28,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(jLabel29, javax.swing.GroupLayout.DEFAULT_SIZE, 374,
Short.MAX_VALUE))))).addContainerGap())
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page6Layout.createSequentialGroup().addContainerGap(javax.swing.GroupLayout.DEFAULT_SI
ZE, Short.MAX_VALUE).addComponent(jLabel26, javax.swing.GroupLayout.PREFERRED_SIZE,
313, javax.swing.GroupLayout.PREFERRED_SIZE) addGap(44, 44, 44))
.addGroup(page6Layout.createSequentialGroup().addGap(82, 82, 82)
.addComponent(jLabel30, javax.swing.GroupLayout.PREFERRED_SIZE, 310,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap(javax.swing.GroupLayout.DEFAUL
T_SIZE, Short.MAX_VALUE)));
```

```
page6Layout.setVerticalGroup(page6Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(page6Layout.createSequentialGroup()
.addGap(34, 34, 34).addComponent(jLabel27, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18) .addComponent(jLabel25,
javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(30, 30, 30) .addComponent(jLabel28,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(15, 15, 15) .addComponent(jLabel26,
javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18) .addComponent(jLabel29,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.UNRELATED).addComponent(jLabel30, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap(javax.swing.GroupLayout.DEFAUL
T_SIZE, Short.MAX_VALUE)));

        page[5].setBounds(0, 10, 420, 430);
        body.add(page[5], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[6].setBackground(new java.awt.Color(255, 255, 255));
        jLabel31.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel31.setText("<html><p align=justify>This factor is determines the amount
of influence of one cluster to its neighboring clusters. Its value ranges from 0 to 1.
The larger it gets, the more clusters get closer to each other.");
        jLabel32.setBackground(new java.awt.Color(220, 220, 230));
        jLabel32.setText("   C.  Distance Metric ");
        jLabel32.setOpaque(true);
        jLabel33.setBackground(new java.awt.Color(230, 220, 230));
        jLabel33.setText("   D.  Initial Learning Rate ");
        jLabel33.setOpaque(true);
        jLabel34.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel34.setText("<html><p align=justify>Distance metrics are used to
determine the dissimilarity of two data points. In this tool, three commonly used
formulas may be used. First is the Manhattan Distance wherein the dissimilarity is
determined by the summation of the distance between the paired coordinates of the
points. Next is Euclidian Distance which is the most commonly used and is equal to the
square root of the sum of squares of the difference between each paired coordinates.
Lastly, the SupNorm Distance uses only the maximum of all the distances between
coordinates as its dissimilarity criterion.");
        jLabel37.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel37.setText("<html><p align=justify>This is the number of times Self-
Organizing Map will train itself in order to discover the actual clustering.
Typically, it is few times as large as the number of points to cluster.  Sufficiently
larger number of iterations will lead to a more actual number of clusters in the data.
");
        jLabel38.setBackground(new java.awt.Color(240, 220, 230));
        jLabel38.setText("   E.  Number of Iterations ");
        jLabel38.setOpaque(true);

         javax.swing.GroupLayout page7Layout = new javax.swing.GroupLayout(page[6]);
        page[6].setLayout(page7Layout);

page7Layout.setHorizontalGroup(page7Layout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page7Layout.createSequentialGroup().addGap(0, 0,
Short.MAX_VALUE).addGroup(page7Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page7Layout.createSequentialGroup().addComponent(jLabel34,
javax.swing.GroupLayout.PREFERRED_SIZE, 346,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(23, 23, 23))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page7Layout.createSequentialGroup().addComponent(jLabel31,
javax.swing.GroupLayout.PREFERRED_SIZE, 339,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(27, 27, 27))))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page7Layout.createSequentialGroup().addContainerGap(25,
Short.MAX_VALUE).addGroup(page7Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false).addComponent(jLabel32, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE).addComponent(jLabel33,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(jLabel38, javax.swing.GroupLayout.DEFAULT_SIZE, 395,
Short.MAX_VALUE)).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page7Layout.createSequentialGroup() .addComponent(jLabel37,
javax.swing.GroupLayout.PREFERRED_SIZE, 341, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(24, 24, 24)))));

page7Layout.setVerticalGroup(page7Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(page7Layout.createSequentialGroup().addContainerGap().addCo
mponent(jLabel32, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jLabel34, javax.swing.GroupLayout.PREFERRED_SIZE, 136,
javax.swing.GroupLayout.PREFERRED_SIZE).addPreferredGap(javax.swing.LayoutStyle.Compon
entPlacement.RELATED, 21, Short.MAX_VALUE).addComponent(jLabel33,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(16, 16, 16).addComponent(jLabel31,
javax.swing.GroupLayout.PREFERRED_SIZE, 51,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18).addComponent(jLabel38,
javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jLabel37, javax.swing.GroupLayout.PREFERRED_SIZE, 66,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap()));

        page[6].setBounds(0, 10, 420, 430);
        body.add(page[6], javax.swing.JLayeredPane.DEFAULT_LAYER);
        page[7].setBackground(new java.awt.Color(255, 255, 255));
        jLabel35.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel35.setText("<html><p align=justify>Once the clustering phase is done,
the tool will automatically display panels for the clusters’ heatmap and the tabulated
summary of results.");
        jLabel36.setFont(new java.awt.Font("Verdana", 0, 10));
        jLabel36.setText("<html><p align=justify>Clustered microarray data are
visualized using a heatmap. It uses different color intensities from green to red to
display the variations of the data points in their range of values.");
        jLabel39.setBackground(new java.awt.Color(220, 230, 250));
        jLabel39.setText("   IV.  Results Visualization and Summary ");
        jLabel39.setOpaque(true);
        jLabel40.setBackground(new java.awt.Color(220, 220, 230));
        jLabel40.setText("   A.  Heatmap ");
        jLabel40.setOpaque(true);

         javax.swing.GroupLayout page8Layout = new javax.swing.GroupLayout(page[7]);
        page[7].setLayout(page8Layout);

page8Layout.setHorizontalGroup(page8Layout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING).addGroup(page8Layout.createSequentialGroup()
.addGroup(page8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(page8Layout.createSequentialGroup().addGap(22, 22, 22)
.addComponent(jLabel39, javax.swing.GroupLayout.PREFERRED_SIZE, 406,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGroup(page8Layout.createSequentialGroup().
addGap(50, 50, 50).addComponent(jLabel35, javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE))).addContainerGap())
```

```
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page8Layout.createSequentialGroup().addGap(0, 0, Short.MAX_VALUE)

.addGroup(page8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).a
ddGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page8Layout.createSequentialGroup().addComponent(jLabel40,
javax.swing.GroupLayout.PREFERRED_SIZE, 378,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap())
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
page8Layout.createSequentialGroup().addComponent(jLabel36,
javax.swing.GroupLayout.PREFERRED_SIZE, 322,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(41, 41, 41)))));

page8Layout.setVerticalGroup(page8Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING).addGroup(page8Layout.createSequentialGroup()
.addGap(81, 81, 81).addComponent(jLabel39, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18).addComponent(jLabel35,
javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(41, 41, 41).addComponent(jLabel40,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE).addGap(18, 18, 18).addComponent(jLabel36,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(120, Short.MAX_VALUE)));

        page[7].setBounds(0, 10, 420, 430);
        body.add(page[7], javax.swing.JLayeredPane.DEFAULT_LAYER);

         javax.swing.GroupLayout backgroundLayout = new
         javax.swing.GroupLayout(background);
        background.setLayout(backgroundLayout);
        backgroundLayout.setHorizontalGroup(

backgroundLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addCom
ponent(header, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(body).addComponent(footer, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE));

backgroundLayout.setVerticalGroup(backgroundLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING).addGroup(backgroundLayout.createSequentialGroup().addCompo
nent(header, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(5, 5, 5) .addComponent(body, javax.swing.GroupLayout.PREFERRED_SIZE, 447,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(footer, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap()));

         javax.swing.GroupLayout layout = new
         javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING).addComponent(bac
kground, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE));

layout.setVerticalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING).addComponent(background, javax.swing.GroupLayout.PREFERRED_SIZE, 520,
javax.swing.GroupLayout.PREFERRED_SIZE));
```

```
        pack();
        this.setLocationRelativeTo(null);
    }

    private void nextActionPerformed(java.awt.event.ActionEvent evt) {
      for(int i=0; i<8; i++){
             if(page[i].isVisible()){
                page[i].setVisible(false);
                page[i+1].setVisible(true);
                if(i==7)next.setEnabled(false);
                else if(i==0)back.setEnabled(true);
                break;
            }
      }
    }

    private void backActionPerformed(java.awt.event.ActionEvent evt) {
      for(int i=8; i>0; i--){
             if(page[i].isVisible()){
                page[i].setVisible(false);
                page[i-1].setVisible(true);
                if(i==8)next.setEnabled(true);
                else if(i==1)back.setEnabled(false);
                break;
            }
      }
    }

    private javax.swing.JButton back;
    private javax.swing.JPanel background;
    private javax.swing.JLayeredPane body;
    private javax.swing.JPanel footer, header;
    private javax.swing.JLabel jLabel1,  jLabel10, jLabel11;
    private javax.swing.JLabel jLabel12, jLabel13, jLabel14;
    private javax.swing.JLabel jLabel15, jLabel16, jLabel17;
    private javax.swing.JLabel jLabel18, jLabel19, jLabel2;
    private javax.swing.JLabel jLabel20, jLabel21, jLabel22;
    private javax.swing.JLabel jLabel23, jLabel24, jLabel25;
    private javax.swing.JLabel jLabel26, jLabel27, jLabel28;
    private javax.swing.JLabel jLabel29, jLabel3,  jLabel30;
    private javax.swing.JLabel jLabel31, jLabel32, jLabel33;
    private javax.swing.JLabel jLabel34, jLabel35, jLabel36;
    private javax.swing.JLabel jLabel37, jLabel38, jLabel39;
    private javax.swing.JLabel jLabel4,  jLabel40, jLabel45;
    private javax.swing.JLabel jLabel5,  jLabel50, jLabel51;
    private javax.swing.JLabel jLabel52, jLabel53, jLabel6;
    private javax.swing.JLabel jLabel7,  jLabel8, jLabel9;
    private javax.swing.JButton next;
    public javax.swing.JPanel page[];
}
```

CHAPTER XI: ACKNOWLEDGEMENT

I would like to express my gratitude to the following people whom I asked for help as I did my SP. First is Sir Geoffrey Solano for being a good adviser to me and to most of my blockmates. Thank you for being considerate and understanding especially when it comes to my extra-curricular activities. Also, on behalf of the block, I would like to thank you for your concern about our SP's and for adopting some of us as your advisees.

To Ma'am Sofia Poblador, thank you for being patient in teaching me the statistical concepts. Sorry for disturbing you – asking you many questions – even at the middle of the night. Thank you for uncomplainingly discussing to me those formulas no matter how difficult it was in Facebook chat.

To Dean Alex Gonzaga and Dr. Carrillo, thank you for being accommodating despite of your busy schedule. And to Lester Ghany, thank you for being approachable, too, and for your SP which is somehow related to mine.

To my blockmates who tried to understand my SP, to those whom I detoxified with, to those who thought my SP was cool, to those who pressured me, and to those who thought I would not finish it, thank you very much!

To the tax-payers, thank you for financing my studies.