CompClustTk
Manual & Tutorial

Brandon King

Version 0.1.10
May 13, 2004

Contents

1 Introduction

1.1 Purpose

CompClust provides utilities designed around the needs of biologists to mine microarray data generally. Specifically CompClust focuses on gaining a more quantitative and qualitative understanding of clustering results and the relationships between them. Initially the software has been developed as a set of python modules suited particular well for use both interactively from with in the python interpreter or to write analysis scripts. The generality of these modules has proved to be useful in many applications within our lab for bioinformatics and genomic analysis. Although the most powerful and flexible way to use CompClust is directly via the python command line interface we have constructed CompClustTK as a more familiar graphical user interface (GUI) to many of the most useful analysis tools. CompClustTK focuses on the analysis tools for comparative cluster analysis as described in (**?**).

This tutorial is for CompClustTK specifically. Other tutorials and documentation have been written as an introduction to using CompClust via the python interpreter directly.

1.2 CompClust Background

CompClust is a Python package written using the pyMLX and IPlot APIs. It provides software tools to explore and quantify relationships between clustering results. Its development has been largely built around requirements for microarray data analysis, but can be easily used in other domains.

Briefly, pyMLX provides efficient and convenient execution of many clustering algorithms using an extendable library of algorithms. It also provides many-to-many linkages between data features and annotations (such as cluster labels, gene names, gene ontology information, etc.) These linkages are persistent through data manipulations. IPlot provides an abstraction of the plotting process in which any arbitrary feature or derived feature of the data can be projected onto any feature of the plot, including the X,Y coordinates of points, marker symbol, marker size, marker/line color, etc. These plots are intrinsically linked to the dataset, the View and the Labeling classes found within pyMLX.

1.3 Where to get help?

If you need help with CompClustTk or CompClust visit http://woldlab.caltech.edu/compClust/.

Or you may send e-mail to Chris Hart (hartATcaltech.edu) or Brandon King (kingbATcaltech.edu).

2 Tutorial Background

2.1 Microarray Data

Users unfamiliar with microarrays in general are encouraged to familiarize themselves with the the biology and technology behind the data. For this tutorial we assume the microarray data is gene expression measurements across several different conditions (eg. Different time points, different cell types, different treatments, etc.). The data will be needed to formatted into a gene expression data matrix, where each row is represents a gene and each column represents a differnt condition. A row in that matrix represents a gene expression vector, or the expression measurements for that gene across every assayed condition.

2.2 CompClust Datasets

CompClust Datasets contain vectors of data, which can be loaded into CompClustTk through a simple tab delimited text format.

Take microaray data for example. If you've done an experiment with four time points; hours 1, 2, 4, and 8. The columns of your data set are the time points and the rows are the individual genes (see below).

# Hour 1	Hour 2	Hour 4	Hour 8
0.72	0.56	0.32	0.06
0.01	0.15	0.80	0.73
0.97	0.95	0.91	0.94

Loading the above data set into CompClustTk and then running a Trajectory Summary plot (see section 3.5.2), you would see that the first gene's trajectory (vector) starts high and gets lower of the four time points. Formatting your data into a tab delimited formated similar to the one shown above will allow you to load your own data sets into CompClustTk.

2.3 CompClust Labelings

Adding a new labeling to any data set is fairly easy. All you need to do is make a tab delimited text file with either one row or one column depending on what type of labeling is appropriate. The only restriction is that labeling must be the same dimensions as it's data set.

For example, if you wanted to add a 'Gene Name' labeling to the data set in section 2.2. You would need a row labeling... i.e. one column with three labels to match the three rows in the data set. Below is an example of this labeling.

Gene Name 1
Gene Name 2
Gene Name 3

If you wanted to make your own cluster labeling (group labeling), you would reuse the same label in one or more rows. For example if I wanted to create a cluster labeling which groups Gene 1 and Gene 2 in one group and Gene 3 in another group, I would create the following row labeling.

Cluster 1
Cluster 1
Cluster 2

One may wish to keep around the time point hours as column labeling as well. To do this, create a tab delimited text file with one row as shown in column label below.

Hour 1	Hour 2	Hour 4	Hour 8

One of the beauties of CompClust is you can attach as many labels as you can think of. In CompClustTk you will see dialog boxes asking you to select cluster labelings, which are row labelings which separates your data into groups. And you will be requested for primary and secondary labelings, which are basically arbitrary row labelings which you may wish to attach. For example, when viewing gene expression data in a plot, you may wish to attach a primary labeling of gene names and a secondary labeling of descriptions.

Column labelings currently can only be taking advantage of by using CompClust from Python, but in the future, these features may be exposed the CompClustTk.

2.4 Cho Example Data

CompClustTk uses example data collected from Cho et. al., 1998. Breifly they synchronized yeast cells using a CDC28 temperature sensitive mutant. After releasing the yeast cells from arrest they colllecte RNA from the cells every 10 minuetes as the cells underwent two rounds of cell division. Using affymetrix arrays they assayed the gene expression profile of every gene in yeast during this experiment (Cho et al., 1998). The resulting gene expression matrix has roughly 6000 genes by 17 time points. We provide a subset of this matrix which includes a total of 380 genes that were both selected by the authors to exhibit cell cycle dependency and meet a minimal noise threshold (Hart and Wold, 2004). Hart et. al. 2004 provides a introduction and theoretical basis for these tools and also provides a case study highlighted the types of biological insights that can be gleened from analysis simular to those described here.

3 CompClustTk

3.1 Introduction

CompClustTk was designed to expose the basic functionality of CompClust. The idea is to bring the CompClust analysis environment to the biologist. Previously, a basic knowledge of Python programming was required in order to use CompClust. This is still true for some of the most advanced analysis one may wish to do.

We hope that CompClustTk will simplify learning to use CompClust by allowing the user to use CompClust without knowing any Python. If you find CompClustTk too limiting, there are a few tools which will help you to adjust to using Python along with the GUI to do more advanced analysis. When ever you trigger an action, the Python code you would have used to do the same thing in pure Python will be stored in the 'Analysis History' section (View—Toggle Analysis History).

For those of you who are feeling daring or just don't like GUIs that much, you can use iPython to access the internals of the GUI, including any data sets or labelings which you have loaded.

In the following sections of this tutorial, we will be using the Cho et. al., 1998 Cell Cycling data mentioned above located in your CompClustTk/Examples/ChoCellCycling directory.

3.2 Load Data Set

The first thing we need to do after loading CompClustTk is to load a data set to work with. Click on 'File—Load Data' Set from the menu.

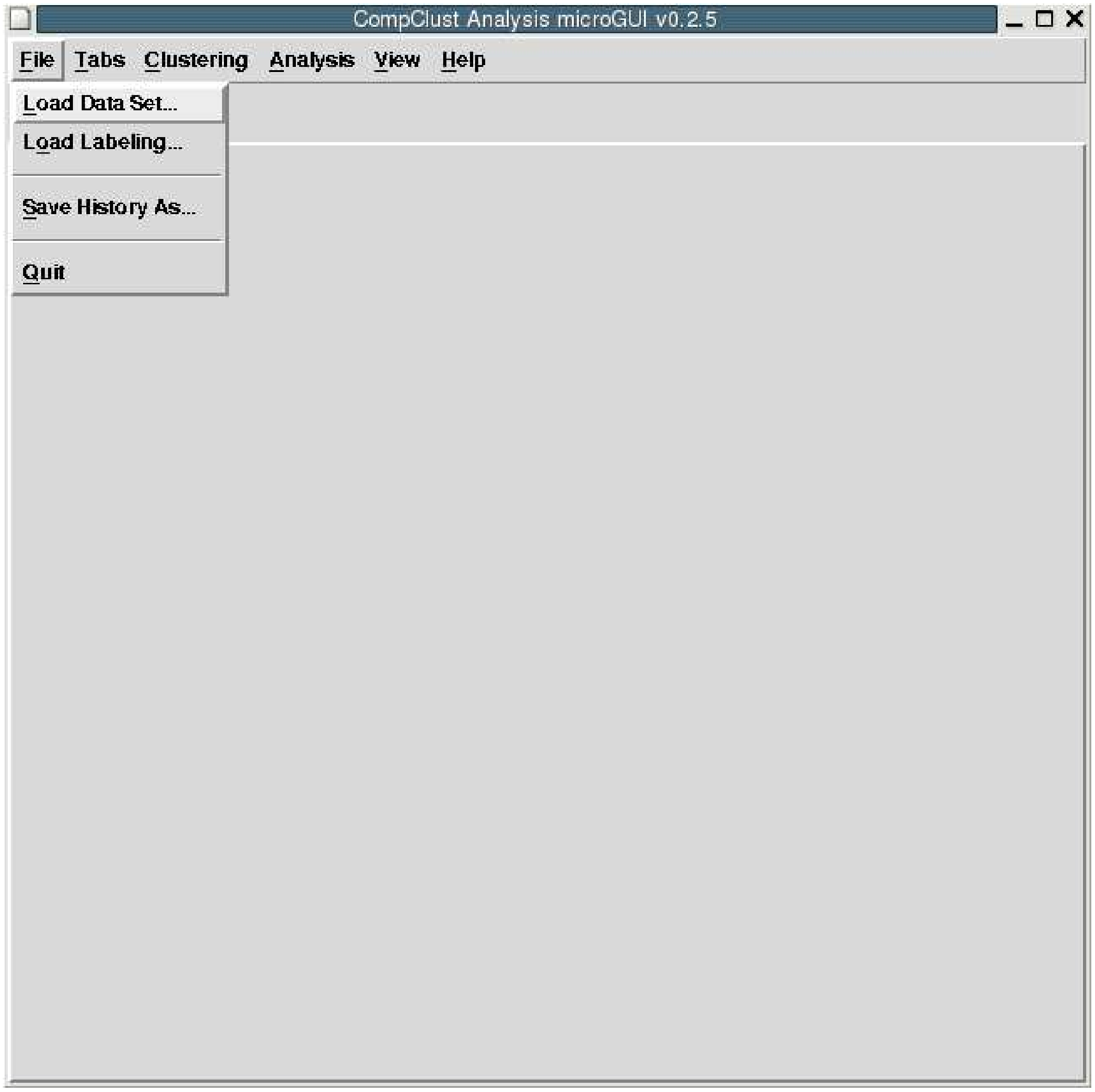


Figure 1: CompClustTk File — Load Data Set

Locate the file named 'ChoCycling.dat' and click 'Open'.

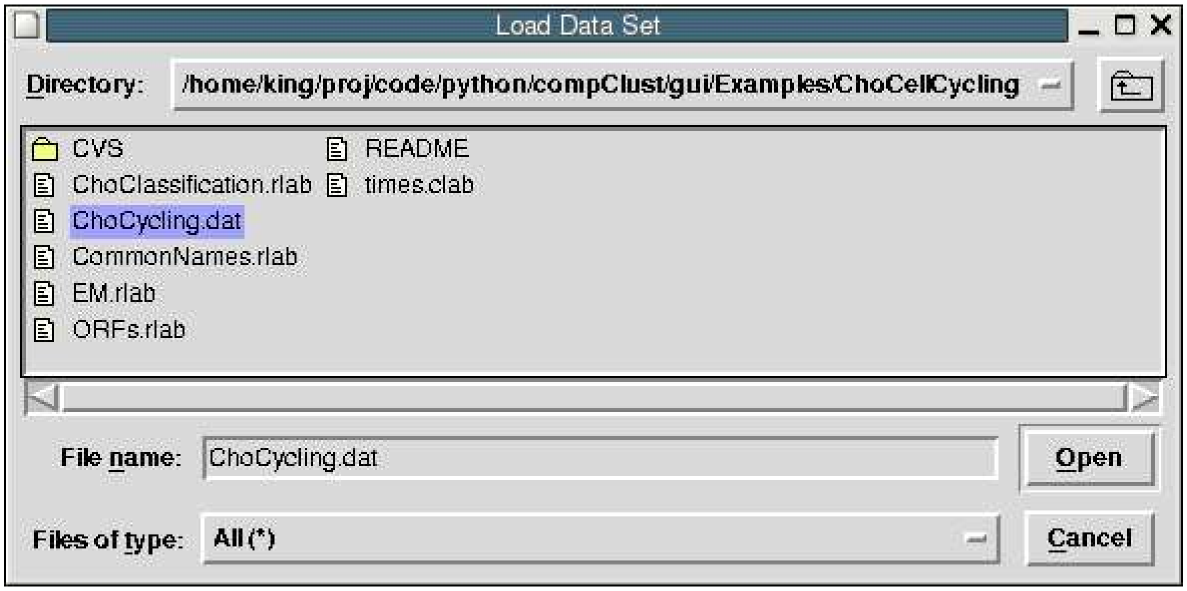
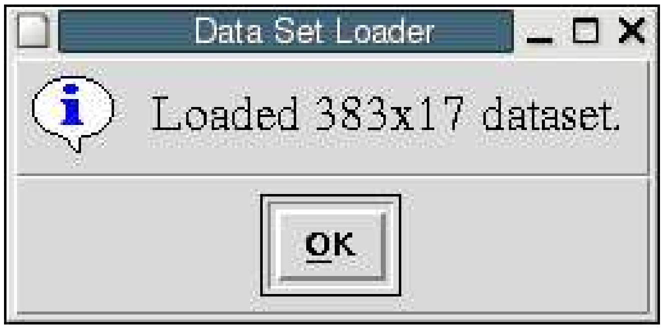


Figure 2: Load Data Set Dialog

If everything went well, you should see a dialog box like the one below telling you the dimensions of the data set you have just loaded. Make sure that the numbers look reasonable, otherwise it may be a sign that your data set was not formatted properly.



Loaded 383x17 dataset.

Figure 3: Load Data Set Complete

3.3 Attach Labelings

Once you have successfully loaded your data set, it will be useful to attach additional information to the data set. The generic name for these annotations is 'Labelings'.

First, let's load the Gene row Labeling so we will be able to know which rows (vectors) represent which genes. Select 'File—Load Labeling' as shown below.

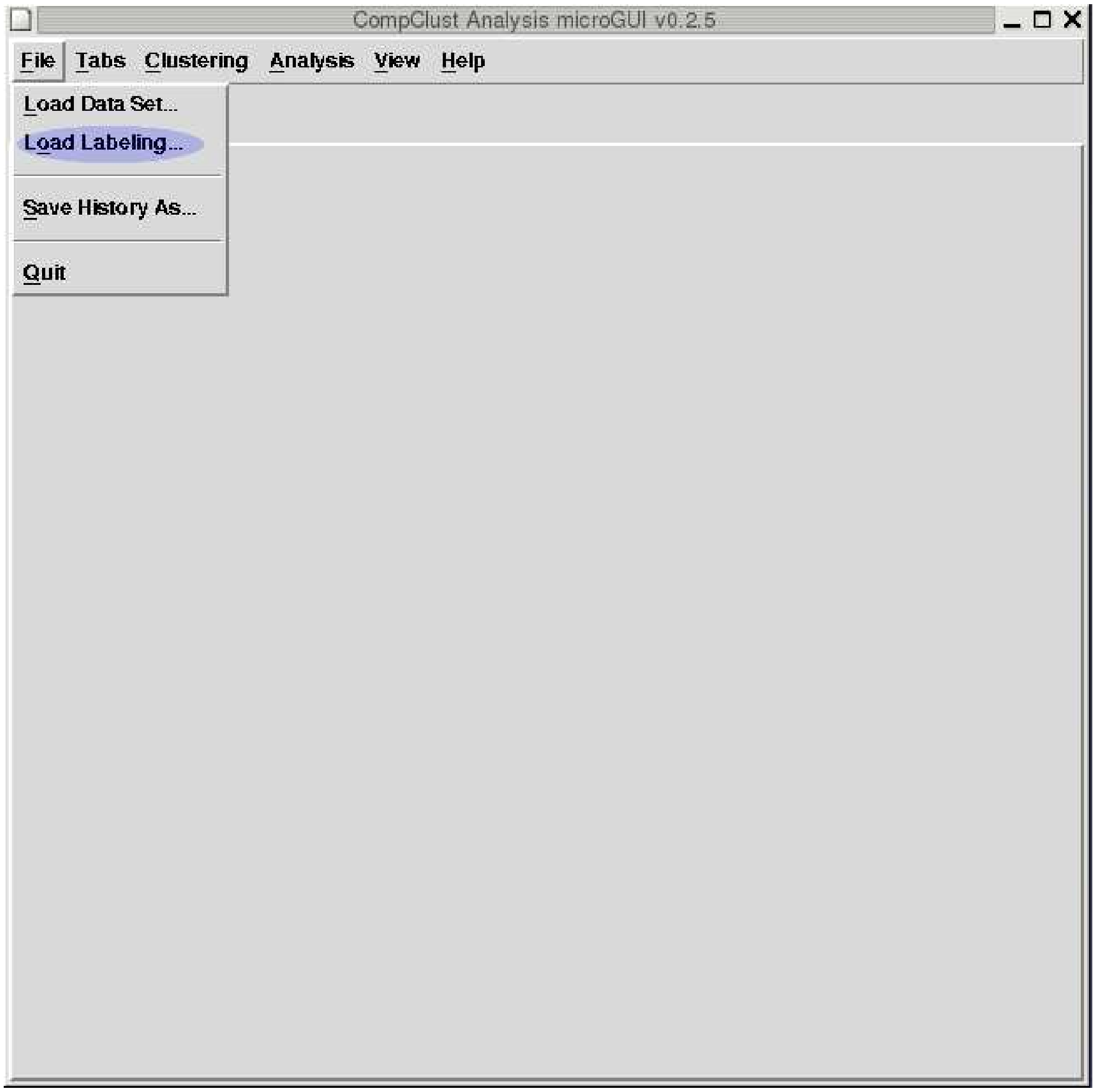


Figure 4: Load Data Set Dialog

The Load Labeling dialog box requires a little more information than the Load Data Set dialog box. Later when you are running an analysis, you will need to select one or more labelings to use, so you should choose a Labeling name which is meaningful. Also, you must tell the program whether you plan to load a row or column labeling (See section 2.3 for more information about CompClust labelings).

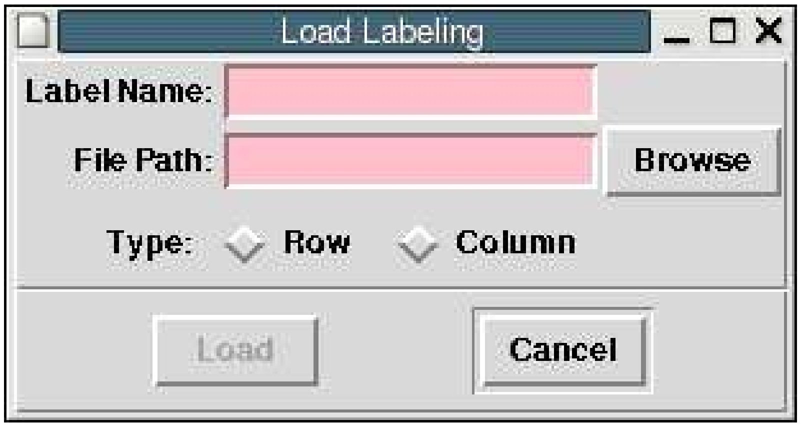
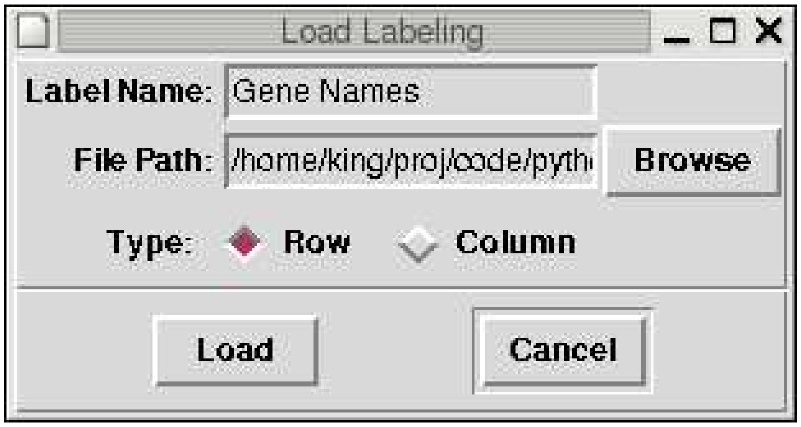


Figure 5: Load Labeling Dialog

Give your Gene Labeling a name like 'blah', press browse and select the file named 'Common-Names.rlab' and then choose the 'Row' radio button as shown below.



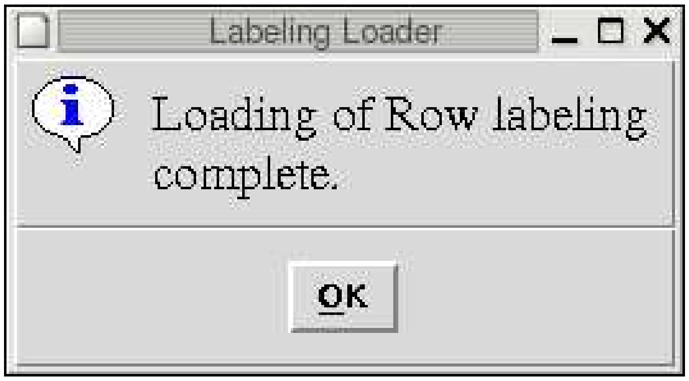
Label Name: Gene Names

File Path: /home/king/proj/code/pyth

Type: [✓ Row] Column

Figure 6: Load Labeling Dialog — Gene Names

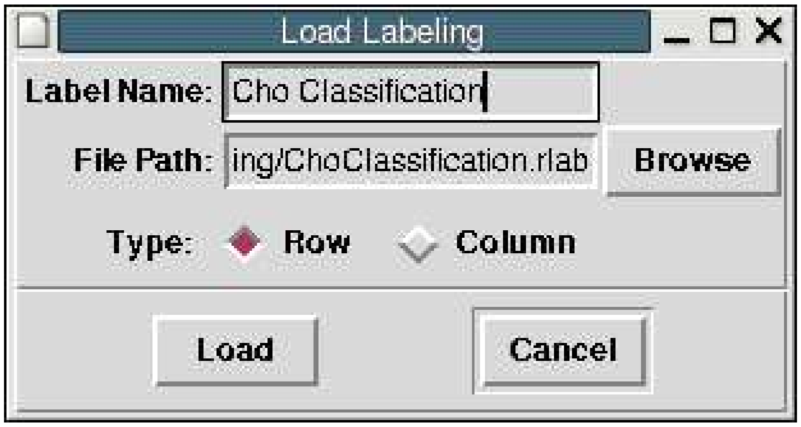
Click 'Load' when you are ready. You should see a dialog box similar to the one below if your Labeling loads successfully.



Loading of Row labeling complete.

Figure 7: Gene Name Load Labeling Complete

Later when we start analyzing our data, we will compare our clustering results (Coming up in section 3.4) to classifications made by Cho. We should attach the Labeling file 'ChoClassification.rlab' as shown below.



Label Name: Cho Classification
File Path: ing/ChoClassification.rlab
Type: [✓ Row] Column

Figure 8: Loading Cho Classification Labeling

3.4 Clustering

3.4.1 Introduction to Clustering

Clustering Algorithms are a general class of unsupervised machine learning techniques which attempt to find an approximation to the optimal partition (or seperation) of a given data set into discrete classes or clusters. An optimal partition is is defined as the partitioning for which each data vector in a cluster is more simular to all other data vectors with the same cluster memberships than to all other data vectors with different cluster memberships. It can be shown that this problem is NP-hard (computationally untracktable, if you think about it, to be certain you have found the ‘”true” optimal clustering you would have to search through all possible clusterings - a very large number as the number of data points becomes big.).

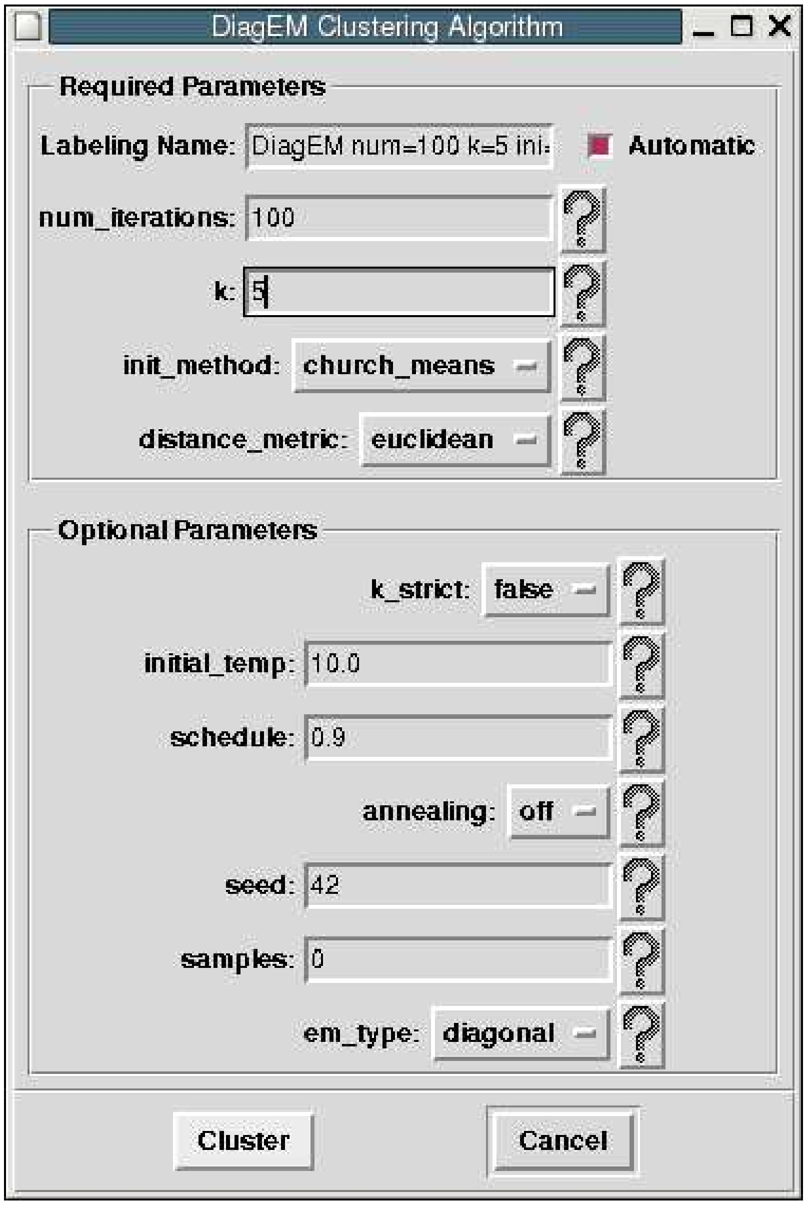
In the context of microarray data analysis clustering has become a staple technique to provide insights into which genes have simular behavior across the conditions being assayed. Clusters essentially form groups of co-expressed genes, using these data as a starting point biologists can then start to address the more interesting questions regarding why these genes are coexpressed. Are the co-regulated functionally, part of simular biochemical pathway, etc. The comparitive tools provide utilities to explore the effects of both what experimental pertabations might have on clustering but also how different algorithms differ from each other on how they partion the genes. This can have potentially profound effects on downstream analysis.

Although we don’t stress it in this tutorial, you can cluster conditions just as easily as you can cluster conditions and the same methods apply.

3.4.2 DiagEM

The first clustering program we will use is DiagEM. We will use most of the default parameters, but we will change the number of clusters we would like DiagEM to create. Change K from two (default) to five as we will compare our results to the Cho Classification, which has been partitioned into five clusters which represent five stages of the yeast cell cycle.

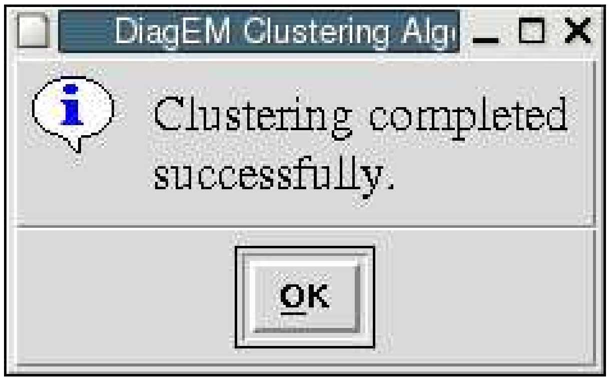
Choose 'DiagEM' from the 'Clustering' menu and then change K from 2 to 5 as shown below.



Labeling Name: DiagEM num=100 k=5 ini
Automatic
num_iterations: 100
k: 5
init_method: church_means
distance_metric: euclidean
k_strict: false
initial_temp: 10.0
schedule: 0.9
annealing: off
seed: 42
samples: 0
em_type: diagonal

Figure 9: Clustering—DiagEM

When clustering is finished you should see a dialog box pop up similar to the one below.

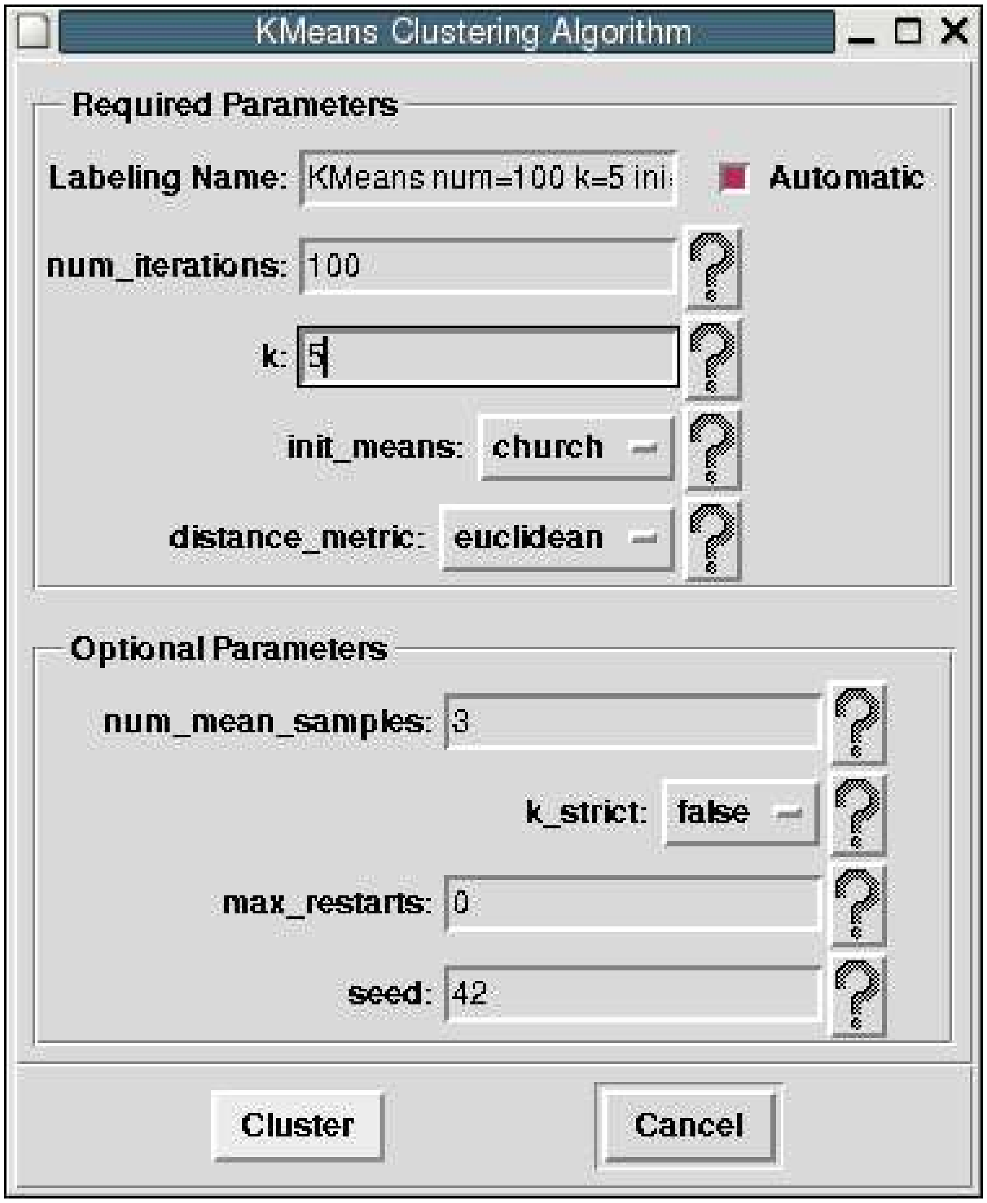


Clustering completed successfully.

Figure 10: Clustering—DiagEM

3.4.3 KMeans

Since every clustering algorithm is different, each one may return different results. We will compare the results of KMeans with DiagEM and Cho's classifications in the analysis section of this tutorial. Select 'KMeans' from the 'Clustering' menu and then change K from 2 to 5. Click 'Cluster' to begin clustering. The KMeans dialog should be similar to the one shown below.



KMeans Clustering Algorithm

Required Parameters

Labeling Name: KMeans num=100 k=5 ini Automatic

num_iterations: 100

k: 5

init_means: church

distance_metric: euclidean

Optional Parameters

num_mean_samples: 3

k_strict: false

max_restarts: 0

seed: 42

Figure 11: Clustering—KMeans

3.5 Analysis

3.5.1 Introduction

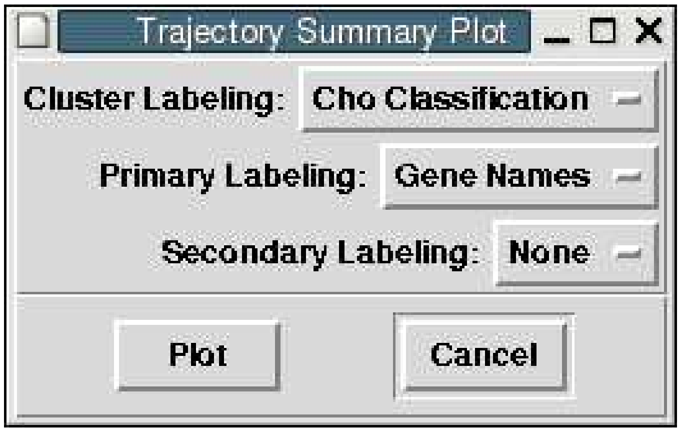
Okay, so we have labelings, both ones that we have loaded, but also labelings created by DiagEM and KMeans. In section 3.5 we will explore the data and view/compare the results of clustering. This will be done with a tool called IPlot which is one of the components of CompClust written by Chris Hart.

At any time if you feel you have too many Tabs open, or you are done with a plot, select 'Close Current Tab' from the 'Tabs' menu.

3.5.2 Trajectory Summary

The Trajectory Summary is a great tool for viewing your data. Given a Cluster Labeling and a Gene Labeling, this tool will allow you to easily visualize and explore your data set.

Let's start by choosing 'Trajectory Summary' from the 'Analysis' menu. To start, select 'Cho Classification' for the 'Cluster Labeling'. So that we know which row (vector) represents which Gene, choose 'Gene Names' for 'Primary Labeling'. The 'Primary Labeling' is the labeling which is displayed when you click on an individual vector. Click 'Plot' when you are ready to view the Trajectory Summary.


Trajectory Summary Plot

Cluster Labeling: Cho Classification

Primary Labeling: Gene Names

Secondary Labeling: None

Figure 12: Clustering—KMeans

Your 'Cho Classification' Trajectory Summary should look similar to the image below. You should have five clusters separating the gene expression data into five stages of the yeast cell cycle. The blue lines are the mean trajectory for a given cluster. The red lines are the standard deviation from the mean.

This is a helpful view to get an idea of what your clusters are doing, but if you want a more detailed view, click on the 'Plot All' button. In this case, lets look at the 'Late G1' cluster. Click on 'Plot All' for the 'Late G1' cluster now.

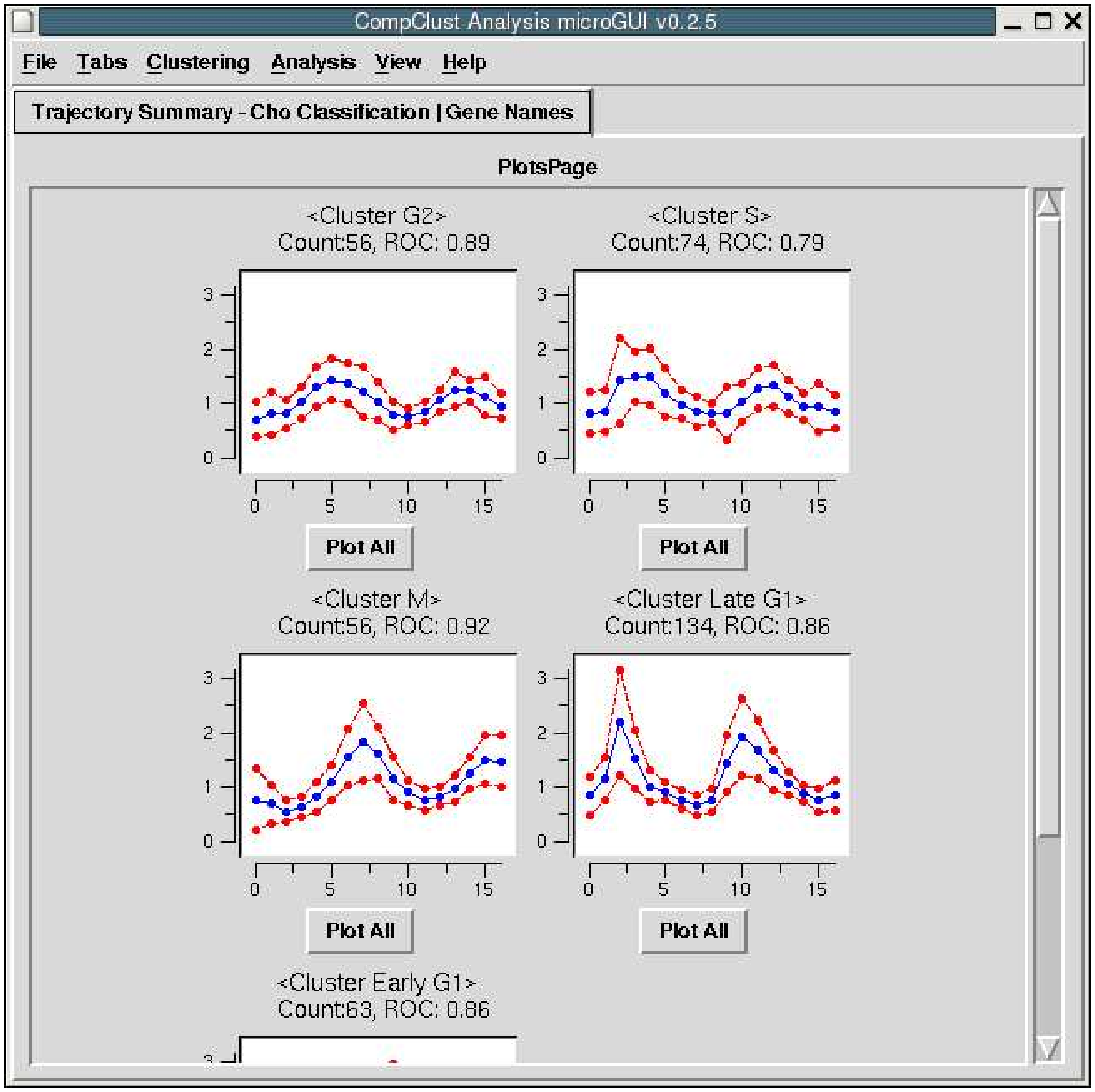


Figure 13: Clustering—KMeans

You should get a plot that looks like the following. The coloring of the trajectories is based on the expression level at time 0 by default. In a future version we may expose the ability to easily change the coloring schema within the GUI itself, but for now if you have the desire to change the coloring, you will have to use the 'Analysis Shell' which is discussed in section 3.6.

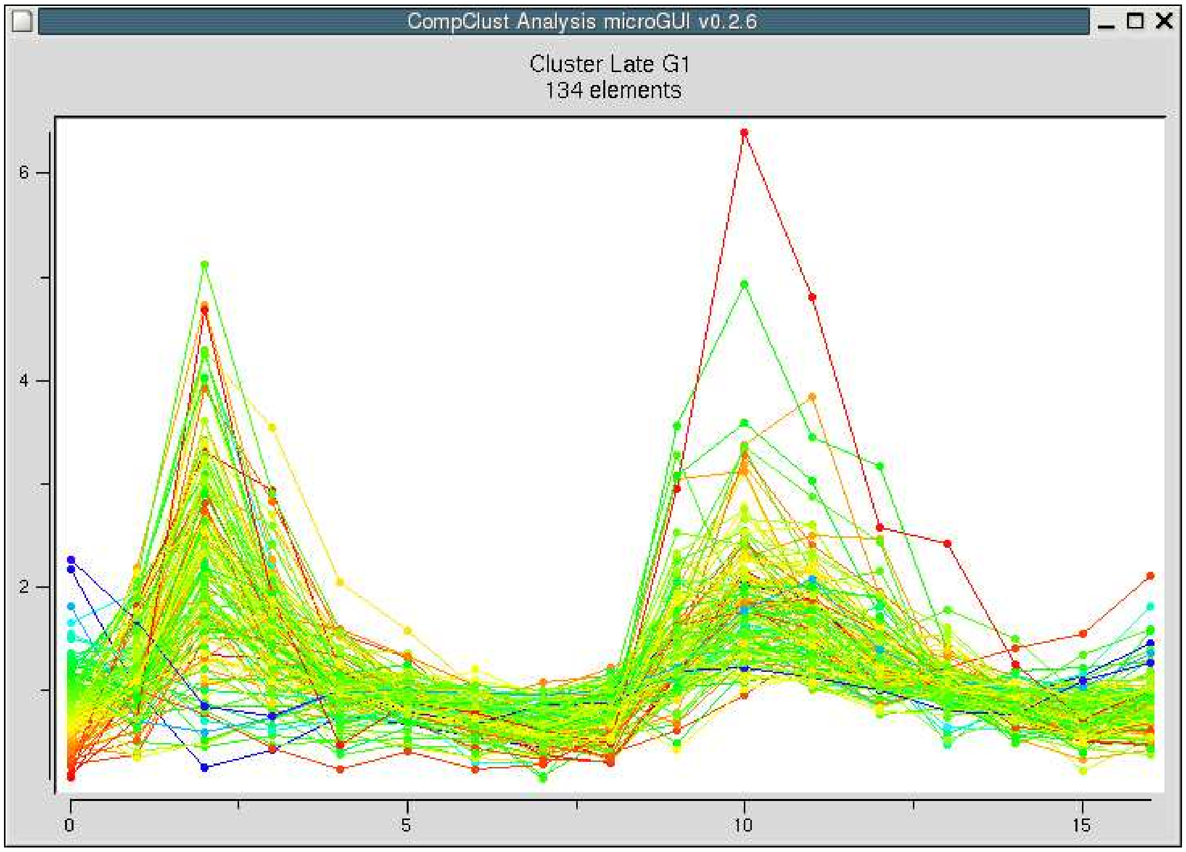


Figure 14: Plot All - Late G1

If you click on any point in the plot, a box in the top right will show up with the text 'Gene Name: (x-cord, y-cord)'. Click on the point shown in the diagram below and you should see 'SCW11: (10.00, 4.92)'. If you control click on the point shown below, a new dialog box appears showing the trajectory for 'SCW11' as shown in Figure 16 on the next page.

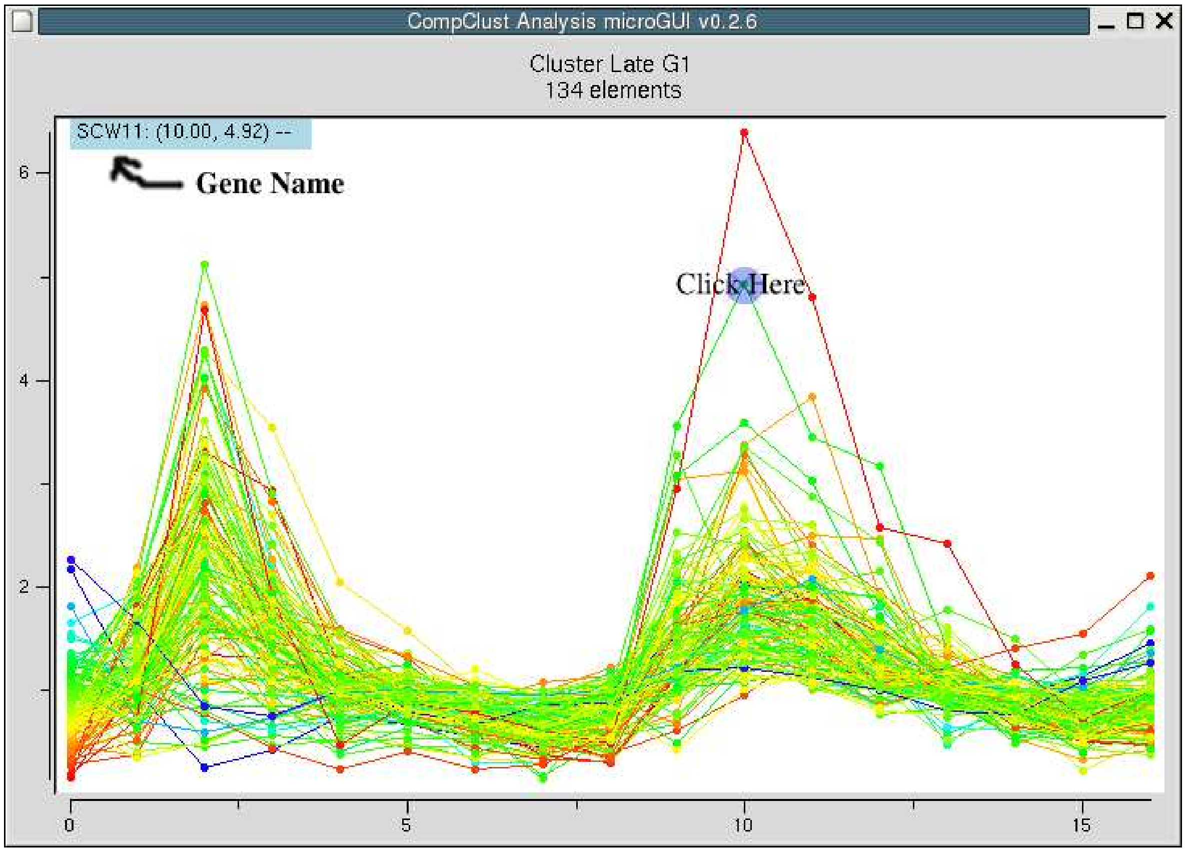
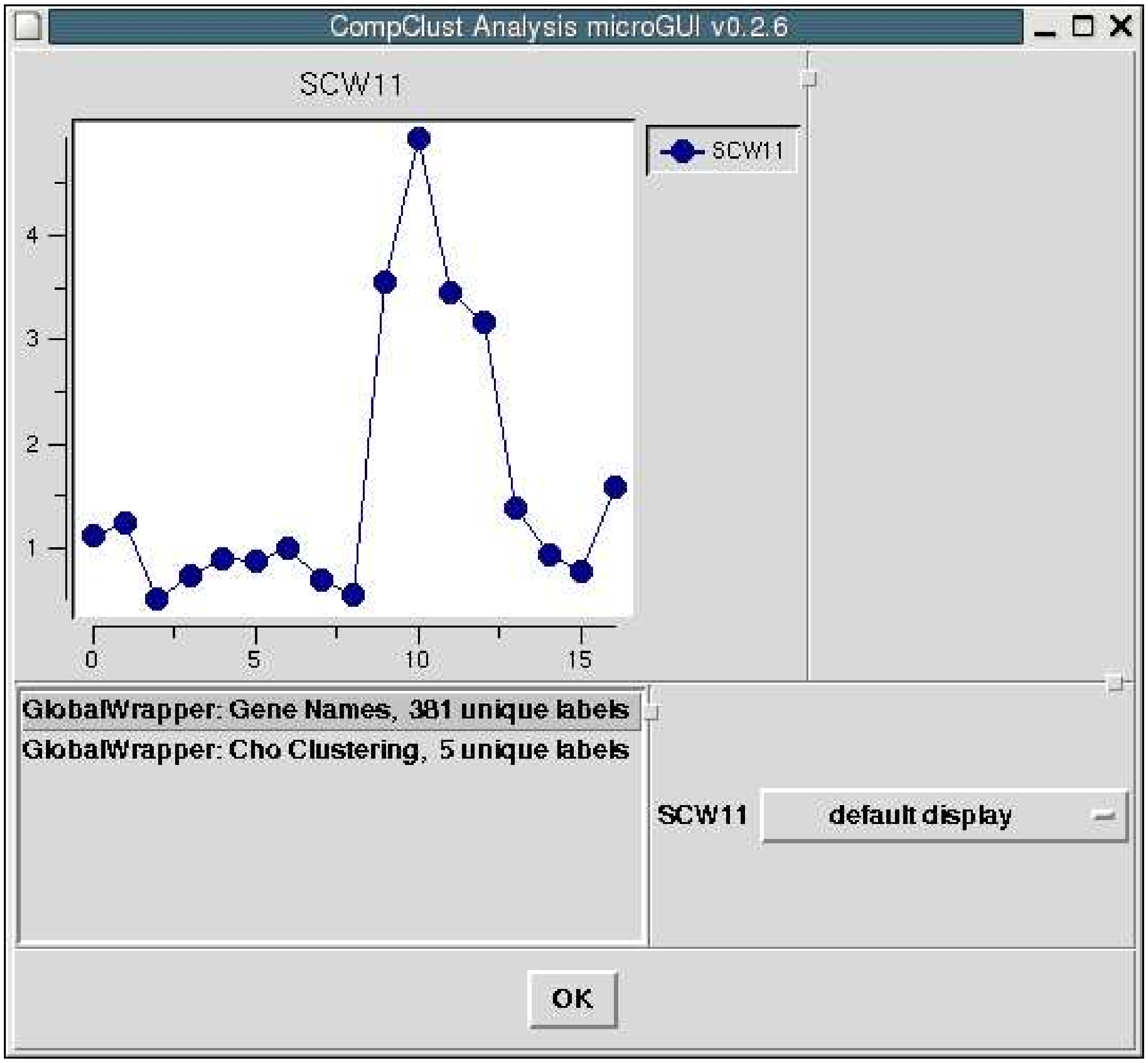


Figure 15: Plot All - Late G1 - SCW11

Note that at the bottom of the figure below, there are two Labelings, 'Gene Names' and 'Cho Classification'. If you click on the 'Gene Names' labeling you should see a pull down menu show up in the bottom right. If you then change this from 'default display' to 'Highlight This Group', this gene will be highlighted in your 'Late G1 - Plot All' display as shown in Figure 18 on the next page.



GlobalWrapper: Gene Names, 381 unique labels
GlobalWrapper: Cho Clustering, 5 unique labels

SCW11 default display

Figure 16: Plot All - Late G1 - SCW11

At this point you can 'Ctrl + Click' on other gene vectors and highlight them as well. This is all we will cover on the Trajectory Summary Plot for this tutorial. Feel free to explore more on your own or continue on to the next section of the tutorial.

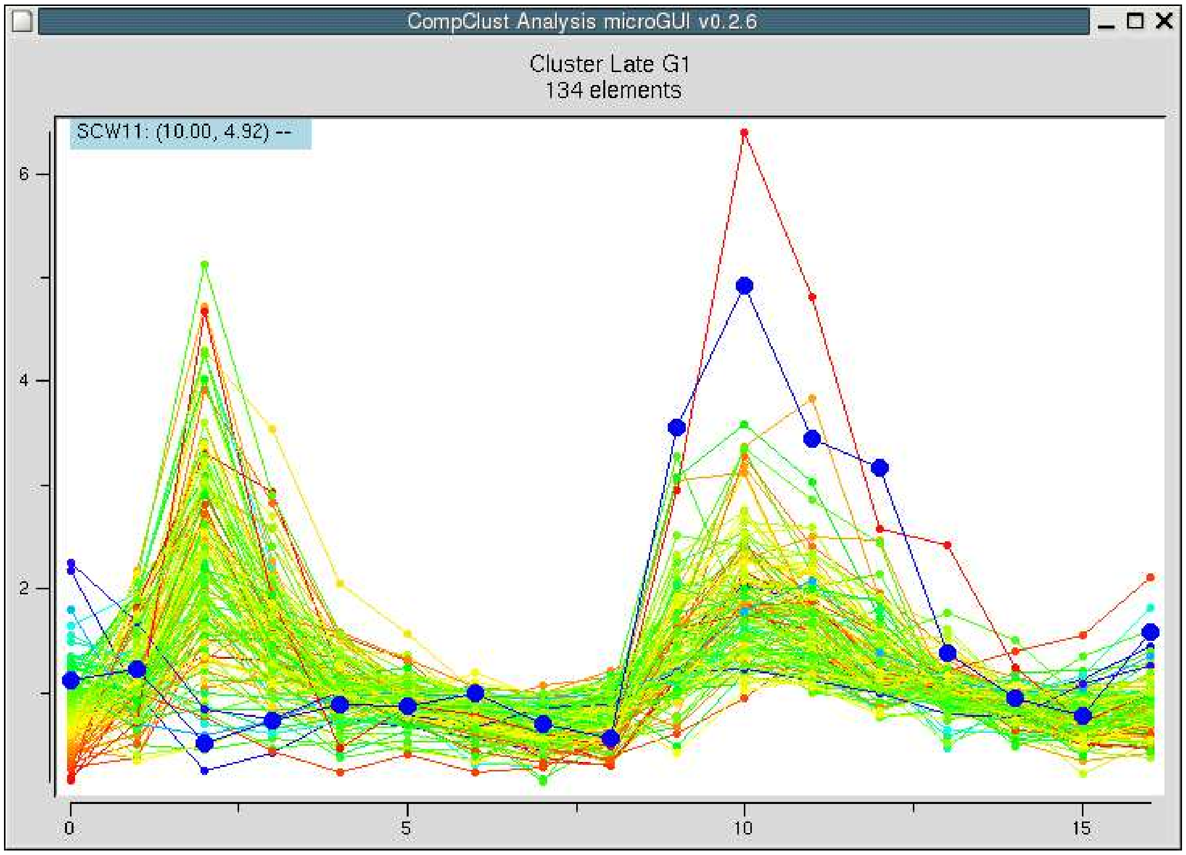
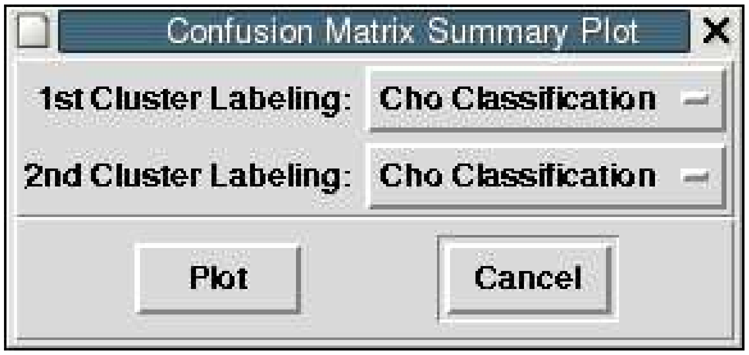


Figure 17: Plot All - Late G1 - SCW11 Highlighted

3.5.3 Confusion Matrix

Now that we have an idea of what our clusters look like from the Trajectory Summary Plots, we will compare the 'Cho Classification', 'DiagEM w/ K=5', and 'Kmeans w/ K=5'. This will be done using a Confusion Matrix Plot.

For starters, let's compare the 'Cho Classification'[1] to itself. Select 'Build Confusion Matrix' from the 'Analysis' menu. Then select 'Cho Classification' for the '1st Clustering Labeling' and '2nd Clustering Labeling' as shown in the figure below. Click 'Plot' when you are ready to move on.



1st Cluster Labeling: Cho Classification

2nd Cluster Labeling: Cho Classification

Figure 18: Confusion Matrix Dialog - Cho vs. Cho

[1]See section 3.3 for information on loading labelings.

You should get a Confusion Matrix plot similar to the following figure. Notice that there are two 'Trajectory Summary' sections being displayed with white backgrounds (top row and last column). Each one of these sections is a clustering, in this case 'Cho Classification' versus itself. If you look at the five clusters in the top row, you'll notice that I have super-imposed green and red bars in the figure below. The green bars are highlighting the number of genes[2] in a given cluster. The red bars are highlighting the name of the clustering.

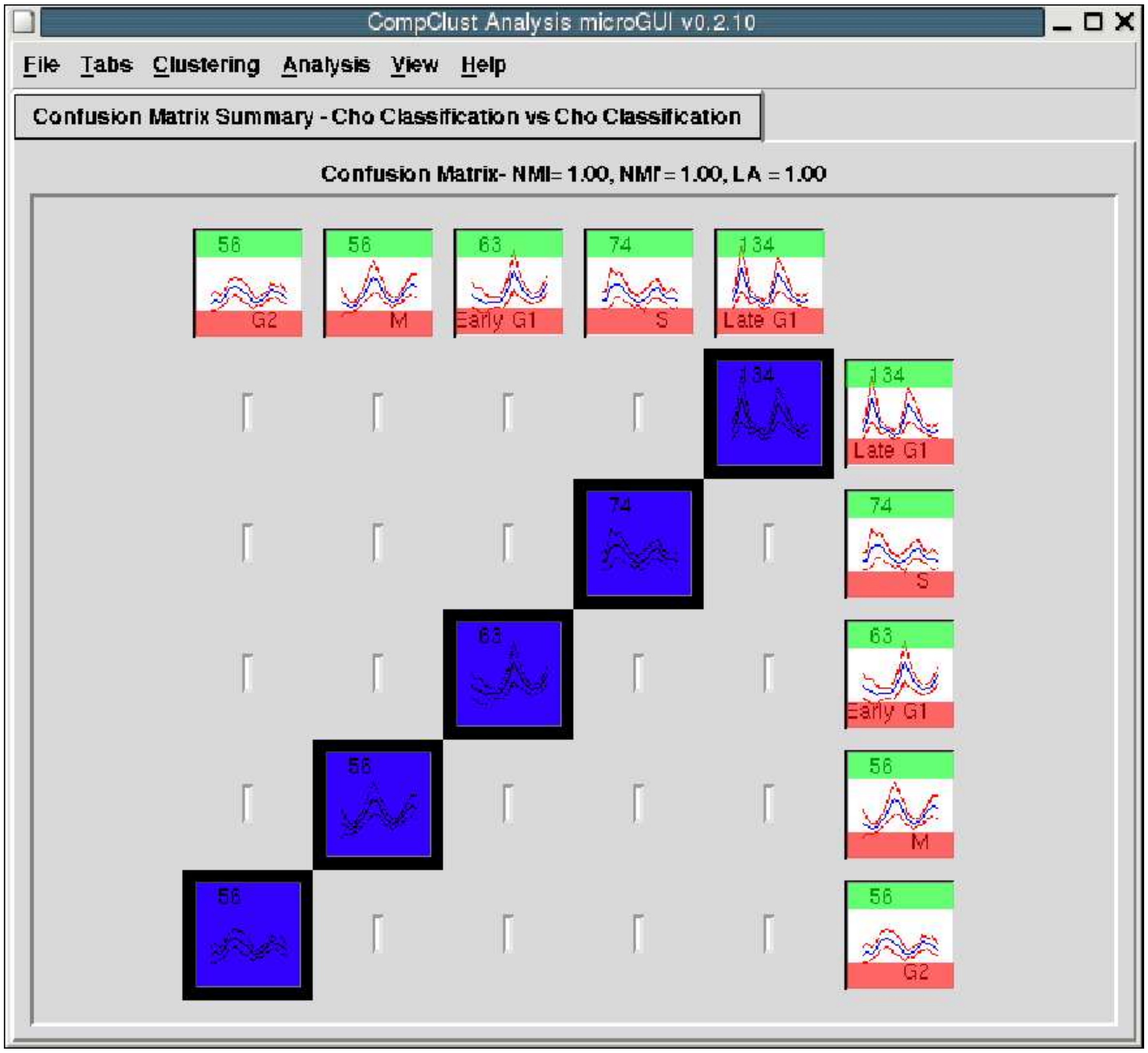

CompClust Analysis microGUI v0.2.10
File Tabs Clustering Analysis View Help
Confusion Matrix Summary - Cho Classification vs Cho Classification
Confusion Matrix- NMI= 1.00, NMI' = 1.00, LA = 1.00
56 G2
56 M
63 Early G1
74 S
134 Late G1
134
74
63
56
56

Figure 19: Confusion Matrix Tab - Cho vs. Cho

[2]CompClust is capable of supporting other types of data beyond gene expression data.

What is this matrix telling us? It's showing us the number of members of column Y that are showing up in row X. For example, if we look at column 2 (M Phase Cluster) and compare it to row 2 (S Phase Cluster), we see that the 'M Phase Cluster' has no members that are shared with the 'S Phase Cluster' (see figure below). Later when we compare our clustering results to Cho's, things won't be as clear as this. If you look at row 1 (Late G1) vs column 5 (Late G1) you'll see that 134 out of 134 members are shared between the two clusters (because they are the same cluster).

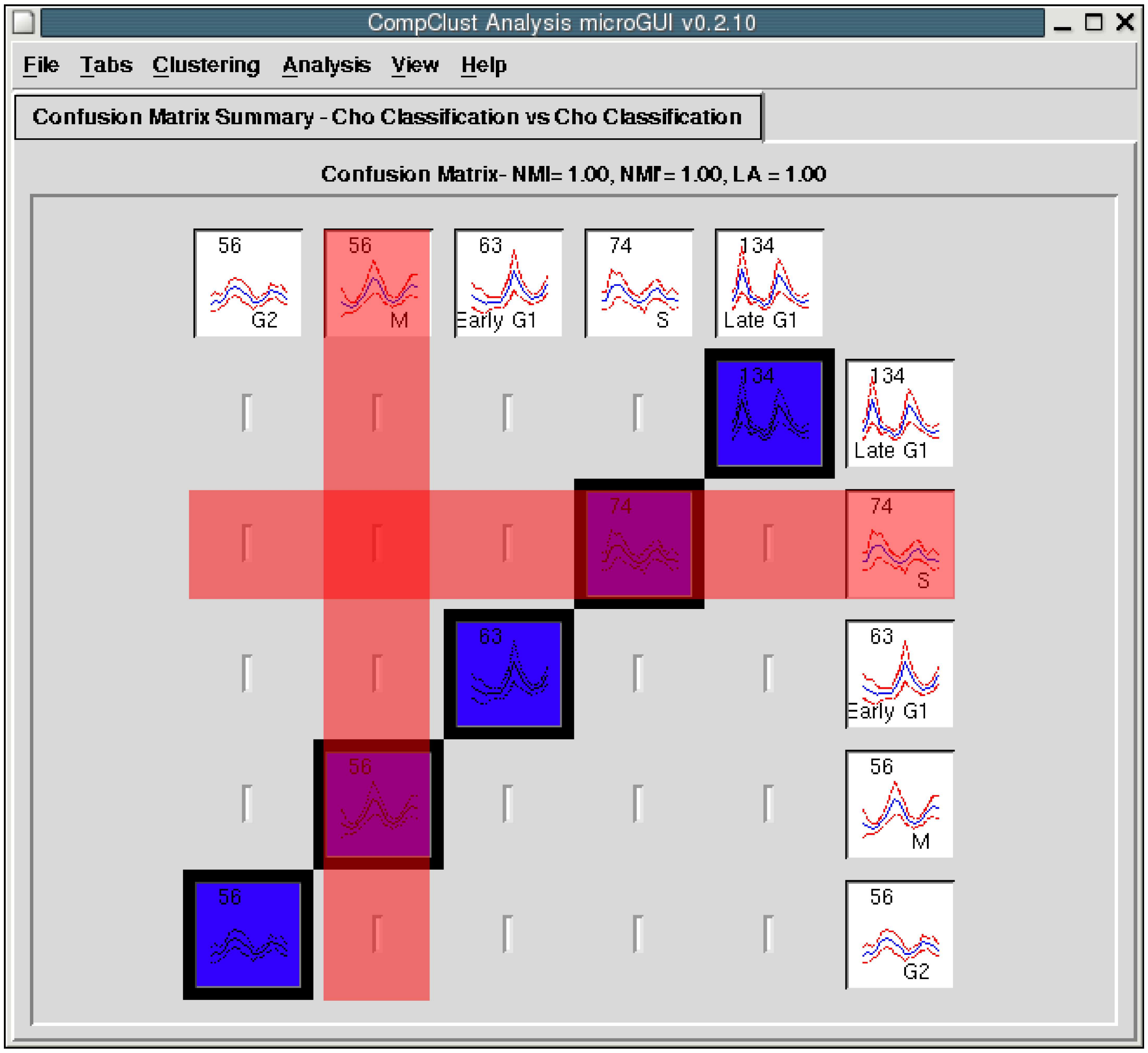
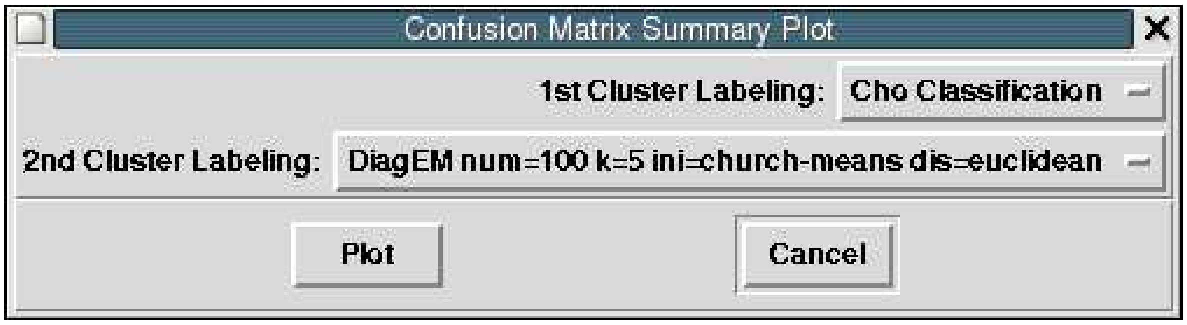

CompClust Analysis microGUI v0.2.10
File
Tabs
Clustering
Analysis
View
Help
Confusion Matrix Summary - Cho Classification vs Cho Classification
Confusion Matrix- NMI= 1.00, NMI'= 1.00, LA = 1.00
56 G2
56 M
63 Early G1
74 S
134 Late G1
134
74
63
56
56

Figure 20: Confusion Matrix Tab - Cho vs. Cho

Now let's move onto a more interesting comparison. Let's compare the 'Cho Classification' to our clustering of 'DiagEM w/ K=5'[3]. Select 'Build Confusion Matrix' from the 'Analysis' menu. Then select 'Cho Classification' for the '1st Clustering Labeling' and 'DiamEM...k=5...' for the '2nd Clustering Labeling' as shown in the figure below. Click 'Plot' when you are ready to move on.



Confusion Matrix Summary Plot

1st Cluster Labeling: Cho Classification

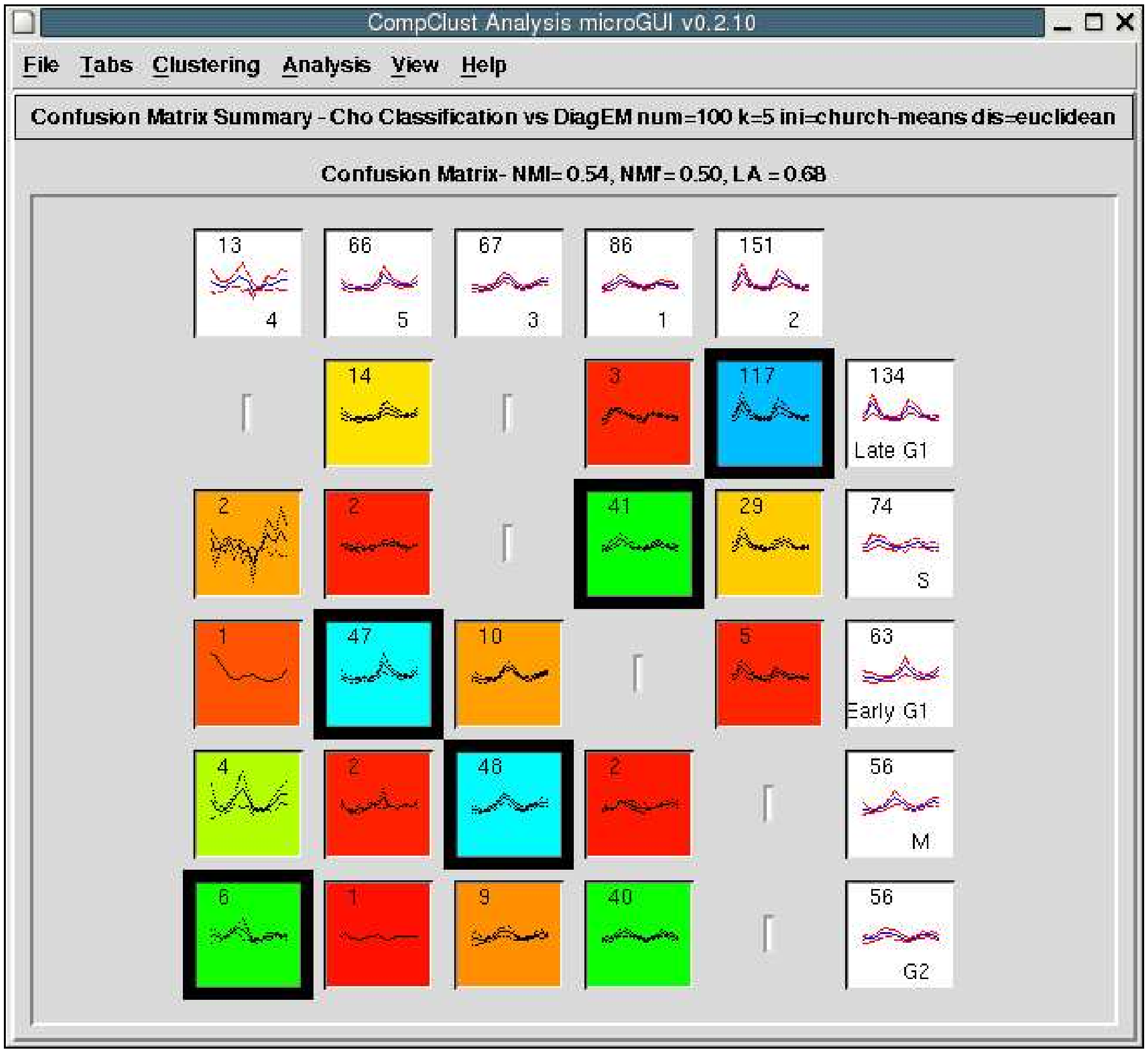
2nd Cluster Labeling: DiagEM num=100 k=5 ini=church-means dis=euclidean

Figure 21: Confusion Matrix Dialog - Cho vs. DiagEM

[3]See section 3.4 if you haven't run DiagEM yet.

As you can see from the figure below, you get a much more interesting plot. We have 'DiagEM' on the X axis and 'Cho Classification' on the Y axis. Note that DiagEM gave the clusters numeric labelings as it had no way of knowing any biologically related information for naming purposes. That would be the job of the user to try to figure that out.

I'll start by describing the coloring scheme. The colors range from red to blue, where pure red is 0% of the members for a given row and column are shared by the two clusters. Pure blue would mean that 100% of the members are shared between the two clusters. If we look at column 1 (DiagEM Cluster #4) and compare it to row 2 (Cho's S Phase Cluster) we see that 2 out of 13 (15.38%) members are shared between the two clusters, which is why it has an orange color. [4]



Confusion Matrix Summary - Cho Classification vs DiagEM num=100 k=5 ini=church-means dis=euclidean

Confusion Matrix- NMI= 0.54, NMI'= 0.50, LA = 0.68

	13 / 4	66 / 5	67 / 3	86 / 1	151 / 2
134 Late G1		14		3	117
74 S	2	2		41	29
63 Early G1	1	47	10		5
56 M	4	2	48	2	
56 G2	6	1	9	40	

Figure 22: Confusion Matrix Tab - Cho vs. DiagEM

[4]Note that all color calculations are based on all the comparisions for a given column. If you were to redo this plot as 'DiagEM' vs 'Cho' instead of 'Cho' vs 'DiagEM', the results will probably be very similar, but the color coding may change significantly. If we looked at the same comparison of 'DiagEM Cluster #4' vs 'Cho's S Phase Cluster' in the 'DiageEM' vs 'Cho' plot, the color would be calculated as 2 out of 74 (2.70%), which would make it much more red than it is in our current plot.

If you look at column 5 (DiagEM Cluster #2) vs row 1 (Late G1) on the figure on the previous page, you will notice that 117 members out of 151 are shared between the two clusters.

3.5.4 ROC Analysis

To Be Written

3.5.5 PCA Explorer

To Be Written

3.6 Advanced Analysis - IPython

3.6.1 Python Introduction

To Be Written

3.6.2 compClust Python Package

To Be Written

4 More of CompClust

4.1 Other CompClust Tutorials

The following tutorials can be found at http://woldlab.caltech.edu/compClust/.

4.1.1 A Quick Start Guide to Microarray Analysis using CompClust

”A Quick Start Guide to Microarray Analysis using CompClust” written by Christopher Hart, covers how to use the Python CompClust environment to do microarray analysis. It may give you a better understanding of the IPlot tools (Trajectory Summary, Confusion Matrices, etc.). It will also teach you how to use some of the more advanced features of CompClust which haven’t been exposed to CompClustTk.

4.1.2 A First Tutorial on the MLX schema

”A First Tutorial on the MLX schema” written by Lucas Scharenbroich, covers the MLX schema. If your want use the full power of compClust using python.

5 Acknowledgements

References

Cho, R. J., Campbell, M. J., Winzeler, E. A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T. G., Gabrielian, A. E., Landsman, D., Lockhart, D. J., and Davis, R. W. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell*, 2(1):65–73. (eng).

Hart, C. and Wold, B. (2004). In preparation. *In preparation.*