Supplementary Material

Description of the cluster algorithm CA

The cluster analysis program CA clusters points into four clusters and assigns a score to each point based on statistical probabilities. The program, which is written in Perl, uses as input a text file containing the raw data for points to be clustered. The program displays the clusters in a plot on the screen, optionally sends it to a printer, and scores every point. An output file is generated which for each point contains the cluster to which the point belongs, a score of the point for that cluster, and probabilities of that point for each of the four clusters.

The program has two very different parts: First, a heuristic algorithm is applied to initially define the clusters, the output of which is used in the second part to score points based on statistics. This second part of the program actually is an iterative process consisting of three identical iterations. In each iteration, the distribution parameters are calculated, points are assigned probabilities for all clusters, and based on these probabilities, new clusters are formed. These are then used in the next iteration. After the iterations, a score is assigned to each point, based on the calculated probabilities.

The heuristic algorithm starts by determining four extreme data points: a left-lower one, a left-upper one, a right-lower one and a right-upper one. If this fails, the algorithm reports an error that there are fewer than four clusters. To determine the extreme points, we use a recursive algorithm. First, the point of gravity of all data points is determined. This point of gravity serves as the origin of a coordinate system. In each quadrant of that coordinate system there should now be at least one data point. When this is not the case the program defaults to a predetermined value to represent the middle of the four clusters. To determine the extreme data point in one of the quadrants, all data points are combined and the point of gravity for these points is recalculated. Again, this point is used to divide the initial quadrant into four sub-quadrants. This is repeated recursively until there is only one point left or until there is no longer a point left. In the former case, this remaining point is the extreme and in the latter case, the center of gravity of the points in the previous step represents an artificially constructed extreme.

Next, we use these extremes as starting points to determine the center of each cluster. Statistics dictate that the center of a cluster is an area of high density of points, and further away from the center the density decreases. This property is used to determine the center of each cluster. A circle is drawn around an extreme and then for all points in the circle the center of gravity is determined which serves as the center of a new circle. This process is repeated until the circle no longer moves or until we took unrealistically many steps (e.g. 10,000). The latter ends the algorithm because of instabilities. Otherwise, the circles migrate to the areas of high density of points. When this is applied to the four extremes, four different centers should result. If this is not the case, the program ends and states that there are less than four clusters. Note that the radius of the circle, which migrates to the high density area, is a degree of freedom, which required empirical adjustment when the program was run initially. If the radius is too small, the circle may remain at the extreme, and if the radius is too large, all circles may migrate to the same center even though there are four clusters.

The three centers of the upper-left, upper-right and lower-right are then used to determine the mid-points for neighboring clusters. Drawing two lines from the origin to the two mid-points defines three sectors. A fourth area is defined for the lower-left cluster by drawing an ellipse with origin (0, 0) and going through two specific points. These two points are defined by adding a constant value, which is heuristically determined, once to the x-coordinate and once to the y-coordinate of the point of gravity of the no-target control samples used in the assay to measure background fluorescence. The final results of this approach are four sectors as shown in Figure 2. The sector in which a data point is situated now determines the initial assignment of the points to clusters. Note that all this is automated, so no user interaction is required. However, at this time, a graph showing all points and the sectors is shown to the user as a visual control so that the user can easily discriminate a bad initialization in case this ever occurs.

Based on this initial clustering, for every cluster, the point of gravity (m_x, m_y) is determined, as are the standard deviations σ_{xx}, σ_{xy}, σ_{yy}. These parameters together unambiguously define a two dimensional normal distribution. A plot of all points of this distribution with a constant probability would be an ellipse in the graph with origin (m_x, m_y). This ellipse actually shows the standard deviation in each direction. It is worth mentioning that the program does not actually use σ_{xx}, σ_{xy}, σ_{yy},

rather, it determines the axes of the ellipse by recalculating $\sigma_{X'X'}$, $\sigma_{X'Y'}$, $\sigma_{Y'Y'}$ in a rotated coordinate system X', Y' with its origin in the point of gravity until $\sigma_{X'Y'}=0$. To find zero we iteratively change the rotation angle through a simple bisection algorithm. All points are then converted to this new coordinate system before putting them in the normal distribution. A point (X, Y) gets the new coordinates (X',Y') after rotation over an angle θ:

$$X' = X \cos(\theta) + Y \sin(\theta)$$
$$Y' = -X \sin(\theta) + Y \cos(\theta)$$

Through this method, the two-dimensional normal distribution can be considered a simple product of two one-dimensional normal distributions, one with parameters $(m, \sigma) = (0, \sigma_{X'X'})$ and the other with parameters $(m, \sigma) = (0, \sigma_{Y'Y'})$. So in the new coordinate system, the two-dimensional normal distribution can be written as:

$$p(X',Y') = 1/(\sigma_{X'X'} \text{sqrt}(2\pi)) \exp(-X'^2 / (2\sigma_{X'X'}^2)) \; 1/(\sigma_{Y'Y'} \text{sqrt}(2\pi)) \exp(-Y'^2 / (2\sigma_{Y'Y'}^2))$$

The two-dimensional normal distributions can be visualized as four mountains rising in the third dimension out of the figure, with their peaks above the points of gravity. Every mountain has a tail (or fairly flat plateau around it), which stretches to infinity. If we take a cross-section at a certain height (=certain probability), we get an ellipse, the size of which is proportional to the standard deviation. With this visualization in mind, *every* point can be assigned probabilities p_1, p_2, p_3, p_4 by the four distributions, ignoring the initial sector definitions, which are n longer needed.

Once the four probabilities for a point are known, the maximum probability among the four is determined and defines the cluster to which the point will be assigned. After assigning all points to their cluster, we repeat the process of determining the centers of gravity and the standard deviations with the new cluster assignment. Based on the new parameters, the new probabilities for each point are calculated and a new cluster assignment is done. This process is again repeated a third time.

Iterating through this process eliminates possible errors in the initial crude cluster (sector) assignment. The idea behind this iterative process of refined cluster assignment is that the areas of

high density will take the upper hand in determining the distribution parameters so some points that are assigned wrongly on initialization should not disturb the distribution significantly and will migrate to the right cluster.

As a last step in the algorithm, the points are scored. The two highest probabilities are determined and used to calculate the normalized score for every point:

$$\text{score} = \max(p_1, p_2, p_3, p_4) \; / \; (\max(p_1, p_2, p_3, p_4) + 2^{nd}\max(p_1, p_2, p_3, p_4))$$

For example, in the case of a point having equal highest probabilities for two clusters the score is 50%, which is the lowest attainable score. We added one modification to this scoring function: if a point belongs to the extreme tail of all clusters, it is assigned a score of zero.

Scores, probabilities and final cluster assignment are all written to a text file `scores.dat`, which can be manipulated by other programs for further processing, or put in a database, or imported in Excel. Because the probabilities are included in this text file, one can easily modify the scoring criteria of the algorithm in a preprocessing step, or when database queries are made. Given this the reader can easily understand the importance of storing these probabilities along in a database, even when one would only be interested in the actual cluster assignment. For the graphics part, the Perl program invokes the freely available program Gnuplot (http://www.gnuplot.org) that plots the points and shows the cluster assignment with colors/symbols from initialization, through the consecutive refinement steps to the end result. For the interested reader we mention that the Perl program prints out a commands file and a data file, both of which are then used by the plotting program Gnuplot.