Published in final edited form as:
Cytometry A. 2011 January ; 79(1): 6–13. doi:10.1002/cyto.a.21007.

Rapid Cell Population Identification in Flow Cytometry Data*

Nima Aghaeepour[1,2], **Radina Nikolic**[1,3], **Holger H. Hoos**[4], and **Ryan R. Brinkman**[†,1,5]
[1]Terry Fox Laboratory, BC Cancer Agency, Vancouver, British Columbia, Canada
[2]Department of Bioinformatics, University of British Columbia, British Columbia, Canada
[3]Department of Statistics, University of Oxford, Oxford, United Kingdom
[4]Department of Computer Science, University of British Columbia, British Columbia, Canada
[5]Department of Medical Genetics, University of British Columbia, British Columbia, Canada

Abstract

We have developed flowMeans, a time-efficient and accurate method for automated identification of cell populations in flow cytometry (FCM) data based on K-means clustering. Unlike traditional K-means, flowMeans can identify concave cell populations by modelling a single population with multiple clusters. flowMeans uses a change point detection algorithm to determine the number of sub-populations, enabling the method to be used in high throughput FCM data analysis pipelines. Our approach compares favourably to manual analysis by human experts and current state-of-the-art automated gating algorithms. flowMeans is freely available as an open source R package through Bioconductor.

Keywords

flow cytometry; data analysis; cluster analysis; model selection; bioinformatics; statistics

Introduction

FCM can be applied in a high-throughput fashion to process thousands of samples per day. However, data analysis can be a significant challenge because each data set is a multi-parametric description of millions of individual cells. Consequently, despite widespread use, FCM has not reached its full potential due to the lack of an automated analysis platform to assist high-throughput data generation [1–3].

A critical bottleneck in data analysis is the identification of groups of similar cells for further study (*i.e.*, gating). This process involves identification of regions in multivariate space containing homogeneous cell populations of interest. Generally, gating has been performed manually by expert users, but manual gating is subject to user variability [4–6] and is unsuitable for high-throughput data analysis [7]. Several methods have been developed to automate the gating process [8]. flowClust is a model-based clustering approach that models

*This project was supported by a Michael Smith Foundation for Health Research Scholar Award to RRB, a MSFHR/CIHR scholarship to NA, a University of British Columbia's graduate fellowship to NA, by NIH grant 1R01EB008400, and by an NSERC Discovery Grant held by HH. This research has been enabled by the use of computing resources provided by the Western Canada Research Grid (WestGrid) and Compute/Calcul Canada. The authors would like to thank Greg Finak and Raphael Gottardo from the Institut de recherches cliniques de Montréal and Thomas Lumley from the Department of Biostatistics, University of Washington for their comments on an earlier version of this manuscript and, Nishant Gopalakrishnan from the Fred Hutchinson Cancer Research Center for his comments on the flowMeans Bioconductor package.

[†]Corresponding author. rbrinkman@bccrc.ca, Tel: +1-604-675-8132 .

cell populations using a Box-Cox transformation to remove skewness followed by a mixtures of t-distributions [9]. flowMerge [10] extends the flowClust framework by applying a cluster merging algorithm [11] to allow multiple components to model the same populations, enabling it to fit concave cell populations. FLAME [12] uses a mixture of skew-t-distributions to make the model more flexible to skewed cell populations. curvHDR [13] is a non-parametric density-based approach, and therefore is not limited to identifying cell populations based on shape. curvHDR models cell populations based on the curvature of the underlying distribution. However, it requires user-defined parameter values and cannot be applied to more than three dimensional data. SamSpectral [14] uses an spectral clustering algorithm to find cell populations, including non-convex ones. Given the high time and memory requirements of the spectral clustering algorithm, SamSpectral finds cell populations based on representative sub-sampling of the data; however, this can potentially decrease the quality of the gating as some biological information can be lost during the sampling process. SamSpectral also requires user-defined parameter values for each data set of similar experiments.

With the advent of high-throughput FCM analysis, millions of cells can be analyzed for up to 20 markers per sample. For these experiments, the runtime of gating algorithms is a bottleneck of automated FCM data analysis pipelines [8]. The K-means clustering algorithm was the first automated data analysis approaches applied to FCM data [15]. Given a d-variable vector $X_1; X_2; \ldots; X_n$, K-means aims to partition X into $K < n$ sets $S = S_1; S_2; \ldots; S_k$ so as to minimize the within-cluster sum of squares:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{X_j \in S_i} \|X_j - c_i\|^2, \tag{1}$$

where c_i is the centroid or center of S_i estimated by its mean value.

However, the adoption of K-means has been restricted, because it requires the number of populations to be pre-identified, it is sensitive to its initialization, and it is limited to modelling spherical cell populations. To estimate the number of clusters, Pelleg et al. [16] and Hamerly et al. [17] extended basic K-means by using the Bayesian Information Criterion and a normality test, respectively. Voting-K-means [18] tries to achieve a good clustering by running the K-means algorithm with a number of different settings and combining the results using an ensemble clustering algorithm. However, the application of these algorithms for automated FCM data analysis has not been successful since the first two are sensitive to noise, and all three require user-defined parameter values [8, 14].

We have developed a new K-means-based clustering framework that addresses the initialization, shape limitation, and model-selection problems of K-means clustering, and can be applied to FCM data. We extended the flowMerge [10] approach by replacing the statistical model with a faster clustering algorithm. By introducing a new merging criterion, our approach finds non-convex cell populations, and we use a change point detection algorithm to estimate the number of clusters.

Materials and Methods

Initial Number of Clusters

The K-means clustering algorithm relies on users to define the number of clusters (K) to find. Using a predefined number of clusters for all FCM samples is not possible due to intersample variability. We solved this problem by automatically choosing K based on a reasonable maximum. The variants of the K-means algorithm discussed in the introduction

try to estimate the exact number of clusters, and are not suitable for estimating the maximum number of clusters. Using the number of cells as the maximum is also not practical due to high runtime required for merging a large number of cells in FCM experiments (*e.g.*, commonly in the hundreds of thousands). Instead, we use the number of modes found individually in every eigenvector of the data. Using individual eigenvectors makes solving the mode-counting problem practical, but results in overlapping clusters (since some cell populations will be projected on more than one eigenvector and will be counted more than once). These overlapping clusters are later merged.

While many mode-detection algorithms are available, we used an approach based on the work of Duong et al. [19] for mode detection using kernel density estimation, which has has previously proven to be successful on FCM data [13]. Formally, for a *d*-variable $X_1; X_2; \ldots; X_n$ sampled randomly from the density function *f*, the kernel density estimator $\widehat{f}$ is defined to be the mean of *n* Gaussian kernel estimations:

$$\widehat{f}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)}{n \cdot h}, \tag{2}$$

where *h* is the bandwidth selected using Scott's rule [20], and $K(\cdot)$ is the Gaussian kernel function:

$$K(x) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot e^{-\frac{x^2}{\sqrt{2}}}. \tag{3}$$

The gradient of the estimator is:

$$\Delta\widehat{f}(x) = \frac{2}{n \cdot h^2} \cdot \sum_{i=1}^{n} (x - X_i) \cdot K\left(\frac{x - X_i}{h}\right) \tag{4}$$

We then used a simultaneous significance test (based on Bonferroni's correction) to find the regions where the gradient is significantly different from zero [19]. Finally, the number of modes in the data is estimated by the number of times that the gradient changes from positive to negative for every one dimensional projection of the data on the eigenvectors. The K-means algorithm is then initialized with the total number of modes across all dimensions.

Merging

We solved the initialization problem at the cost of finding redundant clusters. To find the correct populations, these clusters must be merged. In addition, to capture non-spherical populations, we allow more than one cluster to model a single population (*i.e.*, nearby clusters are merged).

The merging procedure iterates between the following two steps until all of the points are merged to a single cluster: (1) calculate/update the distance between every pair of clusters; (2) identify and merge the closest pair of clusters.

Distance Metric

Given two populations $X = (x_1; x_2; \ldots; x_N)$ and $Y = (y_1; y_2; \ldots; y_M)$, we want to estimate the probability that the point (in this case, cell) y_i belongs to *X*. The closer y_i is to the center of *X*

(*i.e.*, $\bar{X}$), the more likely it is to belong to *X*. However, the probability also depends on the dispersion of *X*. This can be estimated by the normalized Euclidean distance $\frac{\bar{X}-y_i}{S_X}$, where S_X is the sample standard deviation of *X*. In the multivariate case, the direction in which *X* is spread is also important, so the normalization term should be replaced by the covariance matrix. This results in a distance metric called the Mahalanobis distance. Formally, the Mahalanobis distance between *X* and y_i is defined as:

$$D(X, y_i) = \sqrt{\left(\bar{X} - y_i\right) \cdot S_X^{-1} \cdot \left(\bar{X} - y_i\right)^{\mathsf{T}}}, \tag{5}$$

where S_X is the covariance matrix of *X*.

Based on $D(x; y_i)$, we define a symmetric semi-metric (semi-distance) function between populations *X* and *Y* :

$$D(X, Y) = \min \begin{cases} \sqrt{\left(\bar{X} - \bar{Y}\right) \cdot S_X^{-1} \cdot \left(\bar{X} - \bar{Y}\right)^{\mathsf{T}}} \\ \sqrt{\left(\bar{X} - \bar{Y}\right) \cdot S_Y^{-1} \cdot \left(\bar{X} - \bar{Y}\right)^{\mathsf{T}}} \end{cases}. \tag{6}$$

Estimating the Number of Populations

As long as two clusters are overlapping (*i.e.*, modeling the same cell population), the distance between them will be very small, and these will be merged. After several merging steps, when the remaining clusters are well separated, the distance between the next clusters to be merged is significantly larger than the previous ones, indicating that these likely represent separate cell populations. We implemented a segmented regression algorithm to detect the change point in the distance between the merged clusters. This algorithm divides the data to two subsets based on a given break point and fits a line to each of the subsets. The break point that minimizes the error of this model represents the number of clusters for which the clusters are well separated.

Formally, let *N* be the initial number of clusters, $i = (1; \ldots; N)$ the vector of iteration numbers, $NC = N - i$ the vector of number of clusters at each iteration, and *Dist* the distance between the merged clusters at each iteration. The segmented regression model can be described with the following equation:

$$DistR_{(i,BP)} = \begin{cases} A_1 \cdot NC_{(i)} + B_1, \text{if } NC_{(i)} < BP \\ A_2 \cdot NC_{(i)} + B_2, \text{if } NC_{(i)} \geq BP \end{cases}, \tag{7}$$

where *DistR* is the vector of predicted values for *Dist*, *BP* is the break point at which we are expecting an abrupt change of the distance between clusters, and (A_1, B_1) and (A_2, B_2) are the slope and constant of the regression lines for the points before and after the break point, respectively. The least squares method must be applied separately to each segment to estimate the parameters of each line. Finally, the optimized break point BP_{opt} value that minimizes the sum of squared errors (SSE) can be found using exhaustive search over $BP \in \{2; 3; \ldots; \#Clusters - 1\}$:

$$BP_{opt}=\underset{BP}{\operatorname{argmin}}\left(\sum_{i=1}^{N}\left(Dist_{(i)}-DistR_{(i,BP)}\right)^2\right), \tag{8}$$

Figure 1 shows an example where the change point is in the solution with 6 clusters. An animation demonstrating how the model works on this example is available in the Supplemental Materials.

Evaluation

We compared flowMeans to flowMerge and FLAME, the current state-of-the-art automated gating algorithms. Bayesian Information Criterion (BIC) and Scale-free Weighted Ratio (SWR) were used to determine the initial number of clusters for flowMerge and FLAME, respectively. The comparison was conducted using a computer running Ubuntu LTS 8.04 with a 3.2 GHz Intel Pentium CPU and 3 GB of RAM. For flowMeans and flowMerge, 10 random clustering solutions were used for initialization. To avoid model singularity issues caused by the data transformations, a small uniform noise was added to every event before the analysis by any of the algorithms. Convergence was determined using the default criteria of each software. flowMerge and FLAME both have optional free parameters that the user can use to adjust the behaviour of the algorithm (for example by specifying a threshold for the boundary events). We left these parameters at their default values to study the unsupervised performance of all three algorithms.

Our evaluation of the algorithms was based on comparison against manual analysis by human experts that was performed using a set of two dimensional scatter plots. While several metrics are available for comparison of clusterings [21], we used the F-measure because it has proven to be successful for evaluation of the performance of automated gating algorithms [22]. Let n be the number of data points, C the set of membership labels assigned by the human expert, and K the set of membership labels calculated by the automated algorithm. F-measure is formally defined as:

$$F(C,K)=\sum_{c_i\in C}\frac{|c_i|}{n}\cdot\max_{k_j\in K}F\left(c_i,k_j\right) \tag{9}$$

$$F\left(c_i,k_j\right)=\frac{2\cdot R\left(c_i,k_j\right)\cdot P\left(c_i,k_j\right)}{R\left(c_i,k_j\right)+P\left(c_i,k_j\right)} \tag{10}$$

$$R\left(c_i,k_j\right)=\frac{n_{ij}}{|c_i|} \tag{11}$$

$$P\left(c_i,k_j\right)=\frac{n_{ij}}{|k_j|}, \tag{12}$$

where n_{ij} is the number of points with label $c_i \in C$ that are assigned to $k_j \in K$. The points that the human expert had not included in the analysis (for example outliers or biologically irrelevant populations) were excluded before calculating the F-measure.

We measured the F-measure of every sample and reported the average as a single value representative of the distribution. While the average F-measure value helps to evaluate the overall performance of the algorithm across a dataset, it does not help in understanding how these algorithms differ in the analysis of individual samples. We therefore selected four cases where the F-measure values of one of these algorithms was significantly better for further visual illustration of the performance of each method.

Since FLAME's web-based interface does not provide CPU time measurement, all runtimes were measured as wall-clock time on our reference machine. However, we verified that for flowMeans, the difference between CPU time and wall-clock time never exceeded 200 milliseconds.

Datasets

We used two fully gated datasets to evaluate our approach:

Graft versus Host Disease (GvHD)

GvHD occurs in allogeneic hematopoietic stem cell transplant recipients when donor-immune cells in the graft initiate an attack on the skin, gut, liver, and other tissues of the recipient. FCM was used to collect data on bone marrow transplant patients with a goal of identifying biomarkers to predict the development of GvHD. The GvHD dataset is a collection of weekly peripheral blood samples obtained from 31 patients following allogeneic blood and marrow transplant [23]. Cells were stained for four markers, CD4, CD8b, CD3, and CD8.

Diffuse Large B-Cell Lymphoma (DLBCL)

DLBCL is an aggressive lymphoma that can quickly spread to different parts of the body. Its diagnosis is usually performed via lymph node biopsy. The lymphoma dataset from the BC Cancer Agency consists of 30 randomly selected lymph node biopsies from patients seen between 2003 and 2008 [7]. These patients were histologically confirmed to have DLBCL. Cells were stained for three markers, CD3, CD19, and CD5.

Results

Table 1 shows the average F-measure values for flowMerge, FLAME, flowMeans (using the symmetric Mahalanobis semi-distance function), and flowMeans-Euclidean (using an Euclidean distance function) against expert manual analysis. flowMeans and flowMerge performed similarly on both of the datasets, while FLAME had a lower F-measure. As can be seen from the CDF plots shown in Figure 2 in the supplemental information, these averages are not distorted by the presence of outliers.

Figure 2 shows the number of clusters identified by each of the algorithms and the manual analysis. For the GvHD dataset, the results obtained by flowMeans are the closest to those from the manual analysis, followed by those from flowMerge. The number of clusters identified by FLAME are in a much larger interval. For the DLBCL dataset, again, the results obtained by flowMeans are the closest to those from the manual analysis, followed by those from flowMerge. The difference between the results of flowMerge and flowMeans is smaller in the DLBCL dataset. FLAME typically identifies a quite high number of clusters (10 on average).

Table 2 shows that on average, the runtime of flowMeans was significantly lower than that of flowMerge and FLAME. We next examined whether this difference was due to the time requirement of the clustering method or the model-selection approach. Tables 1 and 2 show

that while calculating the symmetric Mahalanobis semi-distance function increases the time requirement, replacing it with a simple Euclidean distance function decreases the accuracy of the identified populations to less than that obtained by the current state-of-the-art methods. Table 3 shows the runtime of the clustering algorithm used by each of these frameworks for identifying 10 clusters. This demonstrates that flowMeans' simpler clustering model is contributing to the lower runtime as well as its approach for estimating the number of clusters without fitting multiple models. Figure 3 shows the agreement between the F-measure of flowMeans and either flowMerge or FLAME. All F-measure values were in the interval [0:5; 1] (shown in panels (a) and (b)), indicating that flowMerge and flowMeans perform similar to each other, even for outlier samples in the correlation plots. The flagged sample in panel (c) shows the extreme case in which FLAME's performance might be closer to the manual gates than that of flowMeans. In this sample, flowMeans has identified an extra population while FLAME has avoided that at the cost of not identifying one of the manually gated populations. Figure 3 panel (c) shows that the F-measure of these two algorithms is rather close while FLAME is slightly higher. However, in panel (d) (flowMeans' best case) FLAME did not perform equally well, since it found too many sub-populations.

The output of each algorithm for the four outlier samples (marked with red *X*'*s* in Figure 3) is shown in Figure 4, with all other samples compared in the supplemental material. Panel (a) in Figure 4 shows the sample chosen in Figure 3 (a). In this sample, the performance of flowMerge is better than that of flowMeans, since flowMerge identified the four populations found by the human expert, while flowMeans found only three. Panel (b) of Figure 4 illustrates the two out of three biologically interesting populations found by flowMeans; we note that the remaining cluster is also missed by flowMerge, even though it identifies three additional populations. Similarly, panels (c) and (d) in Figure 4 show two other samples for which FLAME performed better than flowMeans and vice-versa.

Discussion

Model-based methods have proven to be successful in automating the FCM gating process [10]. However, the time-requirement of these methods represents a bottleneck in applying them to samples with millions of cells and tens of parameters. The application of simpler models to speed up the population identification problem has not been successful as these algorithms are limited by different factors (*e.g.*, reliance on user-defined parameters or specific shapes of populations). For example, while the K-means clustering algorithm (as a special case of Gaussian Mixture Model (GMM)s with spherical variance constant across clusters) is quick compared to other model based approaches, applying it to FCM data has not been successful, since it is limited to spherical cell populations and relies on pre-defined number of populations. A GMM can handle elliptical populations but has a higher running time, since more iterations are required for fitting it to FCM data, which is generally quite noisy. *t* and *skew-t* mixture models are more flexible with respect to kurtosis and skewness at the cost of further increasing the running time [14]. These models can use model selection criteria to estimate the number of populations; however, fitting multiple models compounds runtime requirements.

Since FCM cell populations are not elliptical, flowMerge allows more than one elliptical component to model the same population. We developed a similar framework to extend the K-means algorithm by merging the clusters that belong to the same population. Using the spherical model of the K-means algorithm, our framework has a significantly lower runtime compared to more flexible but computationally expensive statistical models (*e.g.*, a skew/t-mixture model). Improvements in processing time are an important consideration in high-

throughput data production environments. Savings in runtime also increase as the number of measured parameters increases, as is the trend in FCM technology.

The use of more than one centroid to model the same population enabled our K-means based approach to find non-convex cell populations. However, the initial number of clusters needs to be determined before applying K-means. Choosing the correct number of clusters to initialize K-means is not critical, as long as the number selected is larger than the number of cell populations, since the extra (overlapping) clusters are later merged. We used the number of modes in the data (orthogonally projected on one-dimensional sub-spaces) as an upper bound for the number of clusters. Using one-dimensional projections of the data has the drawback of not finding populations that can only be identified in multiple dimensions. flowMeans addresses this problem, to some extent, by projecting the points on the eigenvectors (instead of individual markers) followed by multi-dimensional clustering. However, this can potentially be improved by designing a multi-dimensional procedure for finding a more accurate upper bound for the number of clusters. Regardless of the specific approach, an important advantage of flowMeans over the current model-based approaches is that it doesn't need to fit multiple models to estimate the correct number of clusters. This, along with avoiding an expensive statistical model, resulted in a significantly improved running time (>20 times on average) compared to the current state-of-the-art model-based gating algorithms, without any decrease in accuracy.

We used the position and shape of clusters to identify candidate clusters for merging. We defined a symmetric Mahalanobis semi-distance function that takes the covariance of the clusters into account for calculating the distance between them. At every iteration of flowMeans, these Mahalanobis semi-distances need to be recalculated for the modified cluster. This recalculation procedure represents a bottleneck in the runtime of our framework. However, Tables 1 and 2 show that replacing it with an Euclidean distance function decreased the accuracy of the predicted populations. One possible approach to preserve accuracy and increase speed would be to use a covariance matrix updating procedure (e.g. [24]) to update the symmetric Mahalanobis semi-metric without recalculating it.

Our empirical evaluation was based on comparison against manual analysis. While a wide range of metrics are available for cluster evaluation, we used F-measure as it has been shown to have a better performance in discriminating between the clustering solutions that are similar or different from the manual analysis [22]. The F-measure values show that flowMeans and flowMerge perform similarly, both on average and for individual samples (distributions of F-measures are shown in supplemental materials). In spite of using a more flexible statistical model, FLAME usually has a lower F-measure. Figure 2 suggests that this might be due to the high number of populations that FLAME identifies. To further study the characteristics of these algorithms, we used the F-measure values to select four extreme case samples where the performance of the algorithms varies significantly for visual comparison. While visual comparison generally confirmed the F-measure values, it is important to note that due to the high dimensionality of the data, the margins of the populations could not be effectively visualized. Using human gates as the gold standard for comparison is also complicated as human results can be subjective and highly variable [4–6]. For example, in Figure 4(d) it is not clear if the human has missed the green population found by flowMeans, has intentionally decided to merge it with the blue population, or has marked those cells as outliers. For cases similar to this, if a sample is critically important and the F-measure value alone cannot be trusted, multi-dimensional visualization (*i.e.*, looking at different bi-variate plots as done in the back-gating procedure) can be used to check the margins using different dimensions. Visualizing cell populations in multiple dimensions remains an area for future

improvement. This includes finding the dimensions (or combination of dimensions) that can effectively visualize the populations using feature selection and feature extraction strategies.

An implementation of flowMeans is publicly available as an R package through Bioconductor, a free, open source and open development software project for the analysis and comprehension of genomic data [25, 26].

Conclusion

We have introduced flowMeans, a fast yet accurate K-means-based automated gating framework. flowMeans addresses all the issues that prevented the application of K-means to FCM data in the past. This makes flowMeans a powerful tool for identification of cell populations as part of high throughput and accurate FCM data analysis.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

References

[1]. Bocsi J, Tárnok A. Toward automation of flow data analysis. Cytometry. Part A. 2008; 73(8):679–680.

[2]. Lugli E, Roederer M, Cossarizza A. Good cell, bad cell: flow cytometry reveals t-cell subsets important in hiv disease. Cytometry Part A. 2010; 77A(2):614–622.

[3]. Chattopadhyay P, Roederer M. Data analysis in flow cytometry: The future just started. Cytometry Part A. 2010; 77A(2):705–713.

[4]. Satoh C, Dan K, Yamashita T, Jo R, Tamura H, Ogata K. Flow cytometric parameters with little interexaminer variability for diagnosing low-grade myelodysplastic syndromes. Leukemia Research. 2008; 32(5):699–707. [PubMed: 17936901]

[5]. Gratama J, Kraan J, Keeney M, Granger V, Barnett D. Reduction of variation in T-cell subset enumeration among 55 laboratories using single-platform, three or four-color flow cytometry based on CD45 and SSC-based gating of lymphocytes. Cytometry Part B: Clinical Cytometry. 2002; 50(2):92–101.

[6]. Van Blerk M, Bernier M, Bossuyt X, Chatelain B, Hautcourt JD, Demanet C, Kestens L, Van Bockstaele D, Crucitti T, Libeer J. National external quality assessment scheme for lymphocyte immunophenotyping in Belgium. Clinical Chemistry and Laboratory Medicine. 2003; 41:323–330. [PubMed: 12705342]

[7]. Hahne F, Khodabakhshi A, Bashashati A, Wong C, Gascoyne R, Weng A, Seyfert-Margolis V, Bourcier K, Asare A, Lumley T, et al. Per-channel basis normalization methods for flow cytometry data. Cytometry Part A. 2009; 77(2):121–131.

[8]. Bashashati, A.; Brinkman, R. A Survey of Flow Cytometry Data Analysis Methods. 2009.

[9]. Lo K, Brinkman R, Gottardo R. Automated gating of flow cytometry data via robust model-based clustering. Cytometry Part A. 2008; 73:321–332.

[10]. Finak G, Bashasharti A, Brinkmann R, Gottardo R. Merging mixture model components for improved cell population identification in high throughput flow cytometry data. Advances in Bioinformatics. 2009:100.

[11]. Baudry, J.; Raftery, A.; Celeux, G.; Lo, K.; Gottardo, R. Combining mixture components for clustering. 2010. In press

[12]. Pyne S, Hu X, Wang K, Rossin E, Lin T, Maier L, Baecher-Allan C, McLachlan G, Tamayo P, Hafler D. Automated high-dimensional flow cytometric data analysis. Proceedings of the National Academy of Sciences. 2009; 106(21):8519.

[13]. Naumann U, Wand M. Automation in high-content flow cytometry screening. Cytometry Part A. 2009; 75:789–797.

[14]. Zare H, Shooshtari P, Gupta A, Brinkman R. Data reduction for spectral clustering to analyze high throughput flow cytometry data. BMC Bioinformatics. 2010; 11(1):403. [PubMed: 20667133]

[15]. Murphy R. Automated identification of subpopulations in flow cytometric list mode data using cluster analysis. Cytometry Part A. 2005; 6(4):302–309.

[16]. Pelleg, D.; Moore, A. X-means: Extending K-means with efficient estimation of the number of clusters. Proceedings of the Seventeenth International Conference on Machine Learning table of contents; San Francisco, CA, USA: Morgan Kaufmann Publishers Inc; 2000. p. 727-734.

[17]. Hamerly G, Elkan C. Learning the K in k-means. Advances in Neural Information Processing Systems. 2004; 17:281–288.

[18]. Kaufman, L.; Rousseeuw, P. Finding groups in data: an introduction to cluster analysis. Wiley; New York: 1990.

[19]. Duong T, Cowling A, Koch I, Wand M. Feature significance for multivariate kernel density estimation. Computational Statistics and Data Analysis. 2008; 52(9):4225–4242.

[20]. Scott, D. Multivariate density estimation: theory, practice, and visualization. Wiley-Interscience; 1992.

[21]. Rosenberg, A.; Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. Proc.of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning; 2007. p. 410-420.

[22]. Aghaeepour, N.; Khodabakhshi, AH.; Brinkman, RR. An empirical study of cluster evaluation metrics using flow cytometry data. Clustering Theory Workshop, Neural Information Processing Systems (NIPS); Whistler, British Columbia, Canada. 2009. http://clusteringtheory.org/papers/empiricalmetrics.pdf

[23]. Brinkman RR, Gasparetto M, Lee SJ, Ribickas AJ, Perkins J, Janssen W, Smiley R, Smith C. High-content flow cytometry and temporal data analysis for defining a cellular signature of graft-versus-host disease. Biology of blood and marrow transplantation : Journal of the American Society for Blood and Marrow Transplantation. 2007; 13(6):691–700. [PubMed: 17531779]

[24]. Igel, C.; Suttorp, T.; Hansen, N. A computational efficient covariance matrix update and a (1+ 1)-CMA for evolution strategies. Proceedings of the 8th annual conference on Genetic and evolutionary computation; ACM; 2006. p. 460

[25]. Aghaeepour, N. flowMeans package at Bioconductor. 2010. http://www.bioconductor.org/packages/devel/bioc/html/flowMeans.html

[26]. Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JYH, Zhang J. Bioconductor: Open software development for computational biology and bioinformatics. Genome Biology. 2004; 5:R80. [PubMed: 15461798]

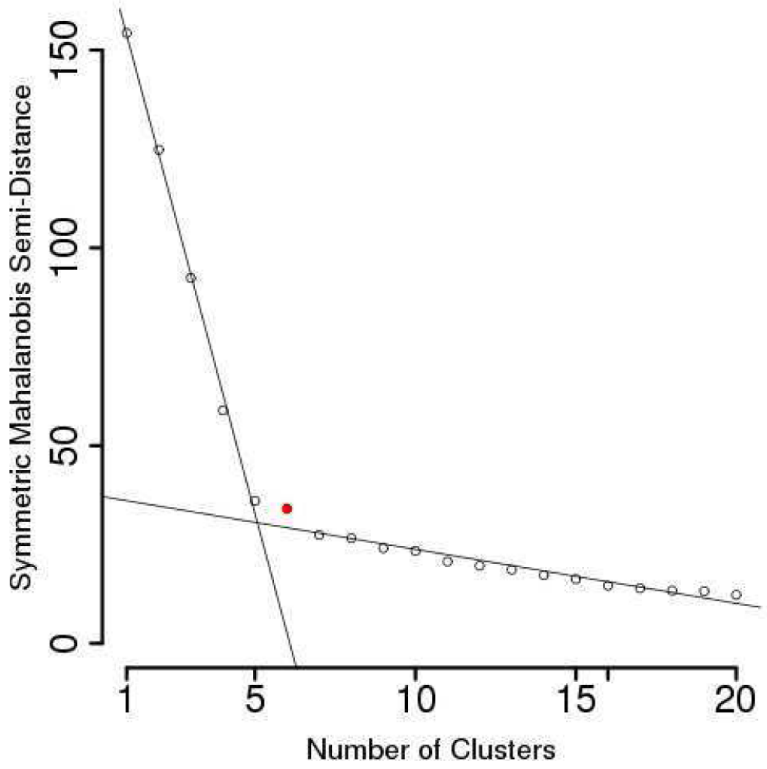

Figure 1.
An example of finding the change point using segmented regression. The chosen solution (shown in red) consists of 6 populations.

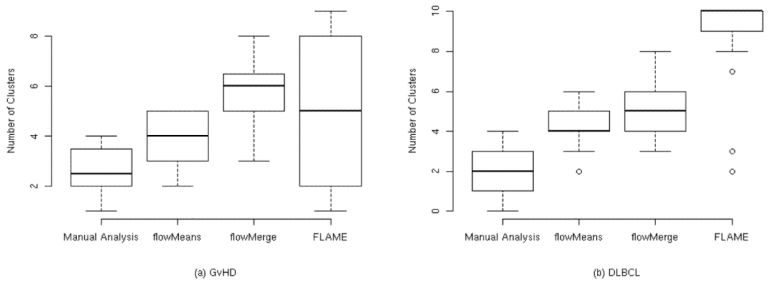

Figure 2.
The number of clusters selected by manual analysis and the three algorithms for the (a)GvHD and (b)DLBCL datasets.

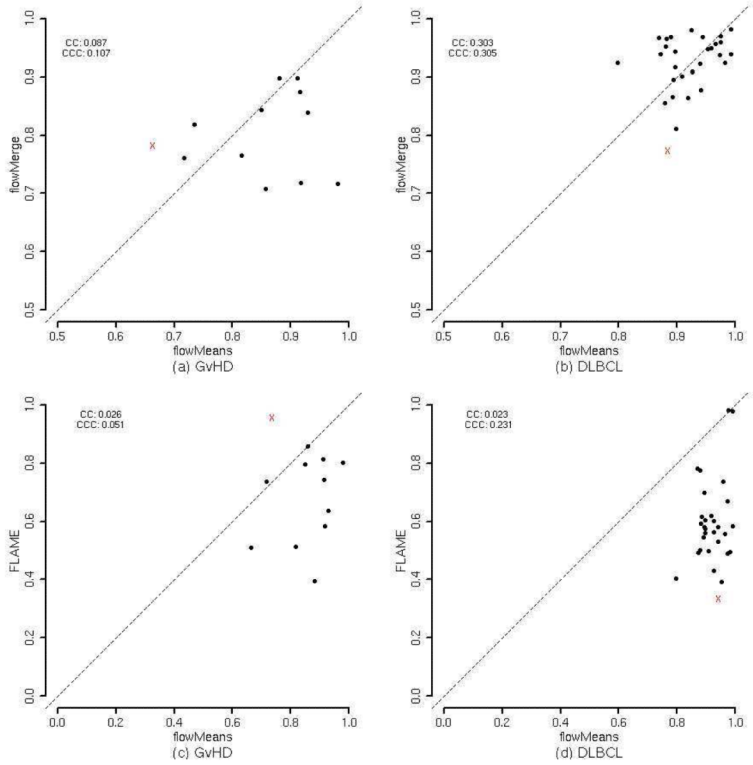

Figure 3.

Agreement between F-measures of flowMeans and either flowMerge(a,b) or FLAME(c,d) on GvHD(a,c) and DLBCL(b,d) datasets. The cell populations for the samples indicated with red *X's* in panels (a)-(d) are shown in respective panels in Figure 4. The dashed line is the agreement line (*i.e.*, $y = x$) that indicates where the performance of the two algorithms is equal. The correlation coefficient (CC) and concordance correlation coefficient (CCC) are shown as legends.

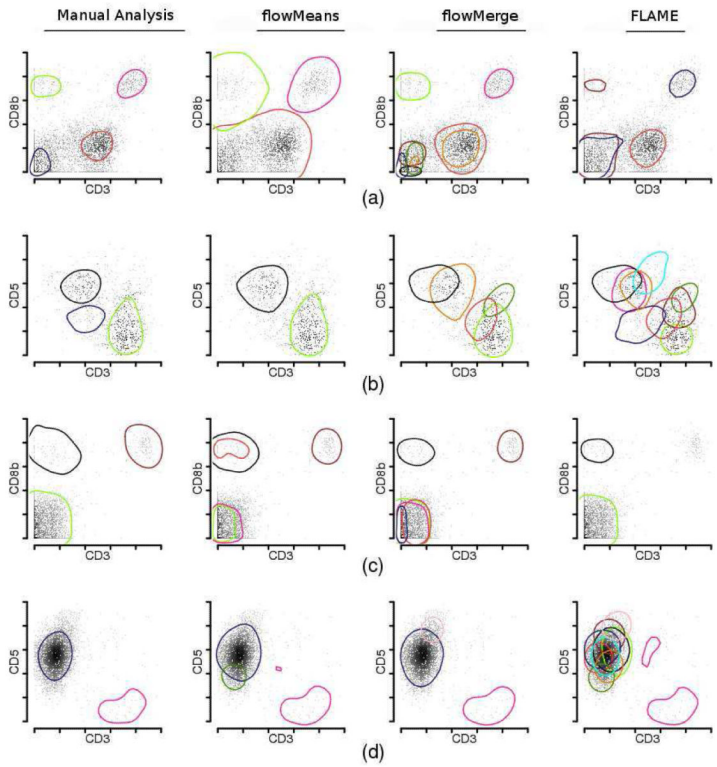

Figure 4.

Panels (a)-(d) illustrate the cell populations found by flowMeans, flowMerge, and FLAME for the samples shown with red X's in respective panels in Figure 3. In this figure, the >90th percentiles of each cluster are visualized to make the boundaries more robust after projection to a two dimensional scatter plot. Therefore the populations might be different from the real distributions on the margins. The pink cluster in panel (d) is a multi-modal population with 2 high-density regions. In every panel, colors of each solution are matched with the solution with the maximum number of clusters.

Table 1

Comparison of F-measure of flowMeans, flowMerge, and FLAME.

Dataset	Mean F-measure (SD)			
	flowMeans Euclidean	flowMeans Mahalanobis	flowMerge	FLAME
GvHD	0.63(0.10)	0.84(0.07)	0.80(0.06)	0.68(0.13)
DLBCL	0.65(0.11)	0.92(0.04)	0.92(0.05)	0.59(0.14)

Table 2

Comparison of Average Wall-Clock (CPU) Runtime of flowMeans, flowMerge, and FLAME.

Dataset	Average Runtime (mm:ss)			
	flowMeans Euclidean	flowMeans Mahalanobis	flowMerge	FLAME
GvHD	00:17	00:28	15:34	18:41
DLBCL	00:13	00:21	11:40	15:35

Table 3

Comparison of Average Runtime of the Clustering Algorithms used for each Framework for Identifying 10 Clusters.

Dataset	Average Runtime (mm:ss)			
	K-means (flowMeans)	Gaussian Mixture Model (flowMerge)	t Mixture Model (flowMerge)	skew-t Mixture Model (FLAME)
GvHD	00:07	04:26	05:37	07:36
DLBCL	00:05	03:31	04:07	05:51