Critical Assessment of Automated Flow Cytometry Data Analysis Techniques

Nima Aghaeepour, Greg Finak,
The FlowCAP Consortium, The DREAM Consortium,
Holger Hoos, Tim R. Mosmann, Raphael Gottardo,
Ryan R. Brinkman, Richard H. Scheuermann

January 23, 2013

Contents

Nature Methods: doi:10.1038/nmeth.2365

Nature Methods: doi:10.1038/nmeth.2365

1 Supplementary Note 1: Algorithm Descriptions

The following descriptions have been provided by the algorithm development groups. These materials have not been modified beyond minor formatting changes. Contact information for further correspondence is provided in Supplementary Table 3.

1.1 FlowCAP-I: Cell Population Identification Challenges

1.1.1 ADICyt

Introduction: The algorithm Automated Detection in Cytometry ADICyt is a hierarchical clustering algorithm specifically designed for identifying relevant cell populations in flow cytometry experimental data. The goal of the algorithm is to reproduce the gating results of a human expert on a general flow cytometry dataset. A cytometrist typically analyzes a flow cytometry dataset by performing a sequence of gating operations in two dimensional projections of the data. ADICyt tries to mimic this process closely.

Method: ADICyt hierarchically (top to bottom) splits the data as would be done in a sequence of manual gating events. At any level of the hierarchical splitting, the algorithm tries to identify the optimal 2D-projection that differentiates the data to subpopulations. The optimality in the above sense is formally defined in the further text. In the following recursive steps of the algorithm, each identified subset of cells is being reanalyzed recursively for further separations (on different 2D projections).

```
int[] ADICyt(data) {
  return ClusterData(data, null)
}
int[] ClusterData(data, last2Dprojection) {
  S := list of all 2D projections of data except last2Dprojection
  C : = empty set of clusterings
  foreach s in S{
    C := C U Cluster2D(data,s)
  }
  C_opt, S_opt := the optimal clustering from C and
  the corresponding 2D projection
  if (Cop_t\$K > 1) # Copt\$K is the number of clusters of Copt {
    K := Copt\$K
    data1,...,dataK := splitData(data, K, Copt\$labels)
    for (i in 1:K)
    {
      if (|data_i| > |fullDataset| * minimalProportionThreshold)
      labels_i := ClusterData(data_i,S_opt)
      else labels_i := {i,...,i}
    }
    labels := mergeLabels(labels_1,..., labels_K)
  }
  else labels := Copt\$labels
```

```
    return labels
  }
```

In the above pseudo code, the function splitData(data,K,labels) generates K datasets, placing the cells with the label i to the i-th dataset. The function mergeLabels($label_{s1}$,..., $label_{s_K}$) combines the input partial labelings to a new labeling such that the number of classes in the result is equal to the sum of those from the partial labelings. The subroutine Cluster2D(data,2Dprojection) returns a clustering of the data on a 2D projection. It is called multiple times throughout the ADICyt execution and it is its sole computation-intensive part. The 2D clustering algorithm used in the current version of ADICyt in place of Cluster2D() is Entropy-Merger, an algorithm motivated by flowMerge [1]. Our experimentations with flowMerge suggest that neither its merging order nor its default decision rule for the final number of clusters is optimal. Therefore, we modified flowMerge and designed Entropy-Merger.

Entropy-Merger starts with a flowClust [2] execution with a number of clusters fixed to a constant based on the size of the data. In the next steps, iteratively a pair of clusters that has the largest relative entropy decrease (RED) is merged together. If all possible pair merges have RED lower than a cutoff, no further merging is performed. The relative entropy decrease is defined as a difference between the original entropy and the resulting entropy, normalized by the size of the smaller of the two merged clusters. The value of RED cutoff is kept on its default value, 1.7, for all the FlowCAP datasets.

Another key part of the ADICyt algorithm is the optimality heuristic for deciding which of the attempted 2D clusterings is optimal or most expert-like. Consider an expert that would be given an anonymous FCS file that she is supposed to gate. She might try to look on all 2D projections of the data and pick one that looks most appropriate for placing the first gate(s). The optimality should mimic this expert decision as much as possible. In ADICyt, the optimality is a function of two entities: The first is the minimal mahalanobis distance between any pair of resulting clusters (this prefers clusterings whose components are well separated). In the second term, the optimality favors those clusterings that have balanced component assignments (in terms of labels entropy). The above two terms are normalized and averaged to form the optimality measure. The optimality of a 1-Clustering is set to 0 so it is inferior to any clustering with at least two clusters

If ADICyt is executed in a mode in which the target number of populations (K_T) is known it runs in a slightly modified mode. As the first step it performs the default calculation.

- If the found number of clusters (K_F) is equal to K_T, ADICyt terminates.
- If $K_F > K_T$, Entropy-Merger is applied with an exception that no RED cutoff is used as a merging stop criterion, but instead the merging is performed until the resulting number of clusters equals to K_T.
- If $K_F < K_T$, the ADICyt algorithm is restarted with the following modification. If at the step of picking copt, the set of clusterings C contains only 1-clusters (no differentiation found), then the set C is being recalculated with iteratively increasing RED cutoff. The RED cutoff is being increased in the sequence 1.9, 2.0, ..., 2.4, until a clustering with at least two clusters is present in C. The increased RED cutoff can

however with higher chance generate an under-merged clustering, a clustering that has more components than an expert would identify. To reduce the rate of false under-merging, the optimality of a clustering that was accepted due to the increased RED cutoff is set to a negative constant, if the labels entropy is higher than 1.0. This causes that the subpopulation identified due to the increased RED cutoff is accepted only if it is a relatively small population; equally sized splitting shall be identified only by the regular RED cutoff. Finally, if the new value of $K_F > K_T$, the merging of the resulting clusters by Entropy-Merger is performed to match the target K_T.

Note that Entropy-Merger requires the probability of cluster assignments for each cell on the input so it can evaluate entropy of a clustering (same as in flowMerge). Thus it cannot be applied directly on a hard clustering (e.g. defined by labels). One can however overcome this technical difficulty by reducing a hard clustering to a mixture model of her choice and execute one M and one E step of the EM algorithm to obtain the probabilities of assignments.

Implementation: ADICyt algorithm is implemented in C# and CUDA and when executed on a computer with a NVidia GPU it runs about 100 times faster than the prototype used for the FlowCAP submission.

Availability: ADICyt algorithm is available in a commercial software available from www.adinis.sk/en/. The software has a typical set of features for analysis of flow cytometry data extended with ADICyt clustering algorithm. A free demo version with limited functionality can be downloaded from the above website.

1.1.2 CDP

We describe the integration of Bayesian non-parametric mixture models, massively parallel computing on GPUs and software development in Python to provide an extensible toolkit for automated statistical analysis in high-dimensional flow cytometry (FCM). The use of standard Bayesian non-parametric Dirichlet process mixture models allows the flexible density estimation of the posterior distribution (MCMC) or modes (EM) of high-dimensional FCM data, and provides a coherent statistical framework for data analysis and interpretation ([3–5]). By exploiting the massively parallel nature of GPUs to achieve greater than 100 fold speed-ups over serial code, it is now realistic to perform large scale data analysis using these methods [6]. To facilitate dissemination to the computational cytometry community, the statistical and computational foundations have been wrapped into fcm, a Python library that makes it simple to design and implement analysis and visualization pipelines for FCM. Used interactively, Python is ideal for rapid prototyping and algorithm development. In addition, Python has robust support for databases, networking, XML processing, GUIs and web development, facilitating the development of full stack applications [7]. Together, statistical mixture models, GPU computing and Python glue provide a principled, efficient and extensible foundation for research into automated FCM analysis [8].

The fcm and cytostream packages can be accessed via:

- `http://code.google.com/p/py-fcm`
- `http://code.google.com/p/cytostream`

1.1.3 FLAME

Introduction: FLAME (FLow analysis with Automated Multivariate Estimation) is an EM-based multivariate finite mixture model algorithm for the analysis of high-dimensional flow cytometry (FCM) data [9]. A distinguishing feature of FLAME is its use of skew-t distributions, which was motivated by the observation that biologically meaningful data clusters are often skew and heavy-tailed. FLAME includes a metaclustering step during which cell populations are matched across samples.

Method: Maximum likelihood estimation via the Expectation-Maximization (EM) algorithm is used to optimally fit the parameters of the mixture model. The algorithm for matching cell populations across samples uses Partitioning Around Medoids (PAM) [10] and linear optimization to achieve a bipartite matching of the population "centers." See [9] for further details.

Implementation: FLAME is written in R and makes use of the EMMIX-skew library, which is written in C. FLAME can run on Linux, MacOs and Windows systems.

Availability: FLAME is publicly available as part of the GenePattern genomic analysis platform developed at the Broad Institute of MIT and Harvard [11]. Detailed information about the use of FLAME in GenePattern is at: `http://www.broadinstitute.org/cancer/software/genepattern/modules/FLAME`.

The EMMIX-skew R package for fitting skew normal and t-densities is available at: `http://www.maths.uq.edu.au/~gjm/mix_soft/EMMIX_R/index.html`.

Source code and documentation for the FlowCAP 1 challenges is available at: `http://www.maths.uq.edu.au/~gjm/FlowCAP/index.html`.

1.1.4 FLOCK

Introduction: FLOCK (Flow Clustering without K) is an automated software system developed for the analysis of high-dimensional flow cytometry (FCM) data [12]. Unlike traditional model-based approaches, FLOCK employs a grid-based partitioning and merging scheme to identify density-based data clusters. The number of clusters is decided based on the density gap between partitioned data regions in different subspaces. FLOCK has been applied to many FCM datasets, including all five FlowCAP datasets in all four types of challenges with encouraging results. The grid-based approach makes the system highly efficient in identifying dense regions in very large data sets. The use of dimension selection and normalization makes the system robust for the analysis of datasets with different number of markers. FLOCK has been used to identify both known and novel cell populations (see [12] for 17 B-cell populations identified by FLOCK). Population statistics are also calculated, including mean fluorescence intensity (MFI), proportions, coefficient of variation (CV), and expression profiles of each population. Populations identified by FLOCK can be mapped based on their centroid positions and compared across samples that use the same reagent panel.

Method: There are five major steps in the FLOCK algorithm: preprocessing, hyper-grid generation, identifying dense hyper-regions, merging dense hyper-regions, and generating final clusters. Preprocessing converts binary FCS files into TXT files, and normalizes marker expressions per-channel to balance their contribution to distance between events.

Hyper-grid is generated through partitioning each data dimension with equal-sized bins. Each hyper-bin may contain a different number of events, which defines the density of the hyper-bin. Dense neighboring hyper-bins are merged together to form the center area of a density-based data cluster. Finally each event in a non-dense hyper bin is assigned to a data cluster based on the Euclidean distance, which finalizes the data clusters(i.e., cell populations).

Implementation: FLOCK is written in C with its user interface implemented in Java. It has been implemented in the publically available Immunology Database and Analysis Portal - ImmPort (`http://immport.niaid.nih.gov`) with an advanced graphical user interface and visualization for open use by the immunology research community, where we are also in the process of linking FLOCK results to cell types defined in the Cell Ontology (CL) [13] for population interpretation and knowledge integration. We have also developed our own open source FCS conversion and transformation method in R (FCSTrans [14]) that generates output consistent with FlowJo (TreeStar, Inc.), to ultimately facilitate an end-to-end free analysis pipeline that starts from instrument FCS files to biologically interpreted cell populations.

Availability: Source code of FLOCK and FCSTrans can be downloaded from: `http://immportflock.sourceforge.net/`. Implementation of FLOCK pipeline with advanced graphical user interface can also be found at `http://immport.niaid.nih.gov/`.

1.1.5 flowClust/Merge

Introduction: FlowMerge is a framework for automated gating of flow cytometry data [1]. It was developed to address the problems with existing clustering methods overestimating the number of cell populations in flow cytometry data.

Method: FlowMerge identifies distinct cell subpopulations in flow cytometry data based on merging mixture components, using an entropy criterion, from multivariate-t mixtures under the BoxCox transformation, implemented in the flowClust package [2]. The cluster-merging algorithm under our framework improves model fit and provides a better estimate of the number of distinct cell subpopulations than either gaussian mixtures or multivariate-t mixtures alone applied to flow cytometry data. The framework allows the automated selection of the number of distinct cell subpopulations and allows for merged cell populations to be readily summarized in a simple manner that integrates with the existing flowClust framework and enables downstream data analysis.

Implementation: flowClust and flowMerge are implemented in R and C, and are available through the BioConductor project.

- flowClust: `http://www.bioconductor.org/packages/release/bioc/html/flowClust.html`
- flowMerge: `http://www.bioconductor.org/packages/release/bioc/html/flowMerge.html`

1.1.6 flowKoh

Introduction: Self-organizing map (SOM) [15] is an artificial intelligence method for clustering, visualizing and analyzing high-dimensional data. It can efficiently handle large datasets and makes no assumptions about underlying dataset distributions. These properties make SOMs attractive for the analysis of flow cytometry (FCM) data. The SOM neural network has a neurobiological background and simple and elegant mathematical model. It is an approach widely used in different domains, yet there were few attempts to apply SOM in the FCM and microarray data analysis..

Method: The flowKoh software is specifically designed and developed for cell population identification in flow cytometry data. It includes the R package kohonen [16] which was applied to cluster three FlowCAP datasets in unsupervised mode. SOM networks are based on competitive learning. The flowKoh algorithm incorporates all the mechanisms that are basic to self-organization: competition, cooperation and self-adaptation. Through iterative selection and learning process data organize itself into two dimensional map, topology preserving. Resulting SOM presents a simplified relational view of a high dimensional data. Visualization of the results indicates that SOM might help us in automatic feature selection. The focus of the current investigation is to determine the accuracy of flowKoh algorithm for cell population identification.

Implementation: The flowKoh software is implemented using R language. It is platform independent and uses R kohonen package and BioConductor flowCore library.

Availability: The algorithm source code is publicly available at `http://commons.bcit.ca/radina_nikolic/docs/flowKoh_R_Code.zip`.

1.1.7 flowMeans

Introduction: The K-means clustering algorithm was the first automated data analysis approaches applied to FCM data. 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. Robust statistical mixture models have been developed to address these issues however, these models increase the time complexity of the algorithms.

Method: 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. Our framework 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.

Implementation: flowMeans is available as a cross-platform R package and have been tested on Linux, MacOS X, and MS Windows.

Availability: The flowMeans software, including documentation and examples, is publicly available as an open source R package through Bioconductor `http://www.bioconductor.org/packages/2.6/bioc/html/flowMeans.html`. A detailed description of the methodology is available elsewhere [17].

1.1.8 FlowVB

Introduction: The increasing dimensionality and size of the data produced by modern flow cytometry platforms poses a major challenge to manual gating. In particular, accurately gating high dimensional data that cannot be directly visualized is difficult. We propose an algorithm that fits a Bayesian mixture model of Student-t distributions to solve the problem of clustering noisy data with an unknown number of clusters.

Method: Mixture models are popular as an automated means of gating flow cytometry data, as mixtures of Gaussian densities, or more robust Student-t densities, can cluster flow data in a statistically meaningful way. A major challenge to the use of mixture models is the requirement of a-priori specification of the number of clusters. If the number of clusters is unknown, we can treat determination of the correct number of clusters as a model selection problem. But this approach can be computationally expensive, requiring multiple runs of the software with varying numbers of clusters specified. We propose a computationally cheaper alternative that allows us to fit a Student-t mixture model (SMM) in a single run using a Variational Bayes (VB) inference algorithm. SMMs have been previously used to analyse flow cytometry data, and generally outperform Gaussian based solutions because of the robustness of the Student-t distribution to outliers. Our contribution is the implementation of an efficient inference algorithm based on [18], which through the use of sparsity promoting priors allows for automatic determination of the number of clusters. In contrast to model selection based methods, we can determine the number of clusters in a single run leading to dramatic decrease in run times.

Implementation: We are offering an implementation of our the algorithm in Python. Minimum requirements to compile and run FlowVB are:

- Python 2.6.5
- Numpy 1.4.0
- Scipy 0.8.0b1
- Matplotlib 0.99.3
- Enthought Traits API 3.4.0
- Cython 0.12.1
- A C-compiler

Availability: The code is freely available from `http://flowvb.github.com/`.

1.1.9 L2kmeans

Introduction: The discrepancy learning process is used to recover the spatial distribution where the individual events, in the given FCS file, are either clumped or scarce. The process provides a quantitative level of information to track the most insiders and outliers.

Method: The insiders are used to determine the number of sub-populations and to use them as centroids initialization for K-means clustering method. Our framework enabled K-means clustering method to be used in high throughput FCM data analysis pipelines.

Implementation: The source code is a framework written in Java language that implements the algorithm design process to actually run on multiple platforms.

Availability: `http://FlowCAP.flowsite.org/download/2010Participants/MEAN_DISCR_CLUST.java`

1.1.10 MM & MMPCA

MM

Introduction: Model based clustering requires serial clustering for all cluster numbers within a user defined interval. The final cluster number is then selected by various criteria. These supervised serial clustering methods are time consuming and frequently different criteria result in different optimal cluster numbers. We developed a new, unsupervised density contour clustering algorithm, called MM (Misty Mountain), that is based on percolation theory and that efficiently analyzes large data sets.

Method: The approach can be envisioned as a progressive top-down removal of clouds covering a data histogram relief map to identify clusters by the appearance of statistically distinct peaks and ridges. This is a parallel clustering method that finds every cluster after analyzing only once the cross sections of the histogram. The multi-dimensional data is first processed to generate a histogram containing an optimal number of bins by using Knuths data-based optimization criterion. Then cross sections of the histogram are created. The algorithm finds the largest cross section of each statistically significant histogram peak. The data points belonging to these largest cross sections define the clusters of the data set.

The algorithm is unbiased for cluster shape, robust to noise and fast (The clustering of 1 million data points in 2D data space takes place within about 15 seconds on a standard laptop PC). It is unsupervised (it does not need estimation for cluster number) and the computation time linearly increases with the number of data points. Its performance with various datasets supports its reliability and utility for automating the analysis of FCM data. More details about the methodology are available elsewhere [19].

Availability: Executable for Windows and example input files are available at Additional files 6-8 in [19].

MMPCA

Introduction: MMPCA (Misty Mountain clustering combined with Principal Component Analysis) is a variation of MM. In the case of MM when the dimension of the data space (i.e.: the number of fluorescent stains) is larger-equal than 5 the data are projected to a 5D subspace by means of principal component analysis. After running MM on the projected data the assigned clusters are projected back to the original data space.

Method:

In the case of MMPCA every data set is projected to a subspace by means of principal component analysis. The subspace dimension is less or equal with the dimension of the original data space. The dimension of the subspace is determined by the analysis of the eigenvalues of the covariance matrix of the data. After running MM on the projected data the assigned clusters are projected back to the original data space.

Availability:

A Windows executable is available at: *FlowCAP-I/Attachments/MM-PCA*

1.1.11 NMFcurvHDR

Introduction: Our core classification method is based on a two-dimensional implementation of an automated method for clustering flow cytomety data, called curvHDR. Given a data set with two dimensions or variables, 2D curvHDR defines N possibly overlapping clusters, where a cluster is defined by a many-sided polygon approximating an enclosed curve. The value of N is determined in a data-driven manner; so N will vary from data set to data set. curvHDR is an attempt to mimic manual gating and human perception of clusters using significant high negative curvature and highest-density region estimation. It is theoretically extensible to any number of dimensions, has a minimum of tuning parameters, and is able to handle data sets of very large size. Another important difference between curvHDR and other methods is that curvHDR is non-parametric, so that it does not have any particular shape restrictions on the gate; this is illustrated in the Results section of the curvHDR paper: Naumann, U., Luta, G. and Wand, M.P. (2010). The curvHDR method for gating flow cytometry samples. Bioinformatics, 11:44, 1-13. A three-dimensional version of curvHDR has since been implemented, and is now available in the R package.

In our R code, the subfunction ClassifyNonoverlapRow is used to assign data points in non-overlapping areas to a cluster. On the other hand, the subfunction ResolveOverlaps is used to assign data points in overlapping areas to the cluster with the closest centroid. Data points which fall outside any clusters are assigned to cluster #0. Ultimately, the main R function ClassifyUsing2DCurvHDR outputs a total of $N + 1$ classifications: N classes corresponding to well-defined clusters/polygons, and one class for data points falling outside any of the N well-defined clusters/polygons.

Feature Extraction Method: In order to use the 2D version of curvHDR, the dimensionality of the data must be reduced to 2. This could have been done in an initial feature extraction step by performing a Principal Components Analysis or an Independent Components Analysis and then keeping only the first two components. However, we thought it would be interesting to use non-negative matrix factorization (NMF) to perform the feature extraction, with the desired number of factors / components set to 2. The reference for NMF is: Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999 Oct 21;401(6755):788-91.

The NMF method requires the data to be non-negative. However, the NDD data set is based on difference data, and therefore has negative values. To enforce non-negativity so that NMF can be used, the absolute values of the NDD were taken. Then, the variances within each column were set to unity, and NMF was performed. Finally, the two-column W matrix output from NMF was transformed using the asinh function, and then fed to curvHDR for classification. The processing stream for the NDD data set was as follows:

$$abs() \rightarrow VarianceNormalization \rightarrow NMF \rightarrow asinh() \rightarrow curvHDR \quad (1)$$

For the other four data sets, if any negative values were detected, the minimum was subtracted from all values to force non-negativity; if there were no negative values, then minimum subtraction was not performed. Thus, subtracting the minimum was conditional on the presence of negative values. (The WNV data has negative values, although it wasnt described as a data set of differences.) The processing stream for the other four data sets was as follows:

$$VarianceNormalization \rightarrow Conditional\ minimum\ subtraction \rightarrow NMF \rightarrow asinh() \rightarrow curvHDR \quad (2)$$

Implementation: Both NMF and curvHDR have been implemented in R, and are freely available as R packages on the CRAN website.

Availability: The R package implementing NMF that we used can be obtained here: `http://cran.r-project.org/web/packages/NMF/index.html`
And the R package implementing curvHDR can be obtained here: `http://cran.r-project.org/web/packages/curvHDR/index.html` A vignette illustrating the use of curvHDR is available at this web page: `http://www.uow.edu.au/~mwand/Rpacks.html`

1.1.12 SamSPECTRAL

Introduction: Spectral clustering is a non-parametric clustering method that avoids the problems of estimating probability distribution functions by using a heuristic based on graphs. Not only does it not require a priori assumptions on the size, shape or distribution of clusters, but it is not sensitive to outliers, noise or shape of clusters; it is adjustable so that biological knowledge can be utilized to adapt it for a specific problem or dataset; and there is mathematical evidence to guarantee its proper performance. However, spectral clustering cannot be directly applied to flow cytometry datasets due to time and memory limitations. To address this issue, we modified spectral clustering by adding an information preserving sampling procedure and applying a post-processing stage. We call this entire algorithm SamSPECTRAL. It has significant advantages in the proper identification of populations with non-elliptical shapes, low density populations close to dense ones, minor subpopulations of a major population and rare populations. In particular, the performance of SamSPECTRAL was tested in identifying a rare population in 34 samples from our stem cell dataset [20]. The rare population in that dataset comprised between 0.1% to 2% of total events and SamSPECTRAL could distinguish it correctly in 27 (79%) samples.

Method: To address the challenge of applying spectral clustering on flow cytometry large data, we developed a data reduction scheme consisting of two major steps; first we sample the data in a representative manner to reduce the number of vertices of the graph. Sample points cover the whole data space uniformly, a property that aids in the identification of both low density and rare populations. In the second step as described in [20], we define a similarity matrix that assigns weights to the edges between the sampled data points. Higher weights are assigned to the edges between nodes in dense regions so that information about the density is preserved. Then spectral clustering is efficiently applied on the resulting smaller graph to identify the clusters. Detailed description of the methodology is available elsewhere [20].

Implementation: SamSPECTRAL is available as a cross-platform R package and have been tested on Linux, MacOS X, and MS Windows.

Availability: The SamSPECTRAL software, including documentation and examples, is publicly available as an open source R package through Bioconductor `http://www.bioconductor.org/packages/2.6/bioc/html/SamSPECTRAL.html`. Also, the algorithm and a report on performance on some flow cytometry datasets are published in [20].

1.1.13 SWIFT

Introduction: Recent advances in data generation platforms and staining reagents for flow cytometry (FC) result in massive datasets containing high-dimensional measurements for millions of cells. The large size, dimensionality, and overlapping nature of cell populations pose a significant challenge to the traditional manual data analysis via 'manual gating' and highlight the need for automated, objective, multivariate clustering. Several statistical model-based flow-clustering methods have been proposed, but they do not scale well to large FC datasets due to their high computational complexity. We propose a scalable clustering framework SWIFT based on weighted iterative sampling (Naim et al., 2010) that scales existing statistical model-based clustering methods to large FC datasets and allows detection of small populations that are frequently of interest.

Method: The proposed clustering algorithm SWIFT has three stages. In the first stage, SWIFT introduces a novel weighted iterative sampling framework for efficient Gaussian mixture modeling by the Expectation-Maximization (EM) algorithm. We show that the proposed weighted iterative sampling, not only increases scalability, but also facilitates better resolution of small clusters. In the second stage (after mixture model fitting), SWIFT examines each of the Gaussian clusters and splits any clusters that are bimodal along any dimensions or principal components. This stage is often crucial for identifying rare populations in high-dimensional flow datasets. Finally in the third stage, a graph-based merging is applied to fuse strongly overlapping Gaussians based on a normalized overlap measure and an entropy-based stopping criterion. This stage allows the method to represent skewed clusters frequently seen in FC data. A major contribution of SWIFT is its scalability to larger datasets. SWIFT provides significant speed-up for model-based clustering methods. Moreover, SWIFT shows excellent results in resolving overlapping and rare populations.

Implementation: SWIFT is implemented in Matlab and requires the Matlab Statistics Toolbox.

Availability: The source code for SWIFT can be downloaded from: `http://www.ece.rochester.edu/projects/siplab/Software/SWIFT.html`. For more detailed description, please see the manuscript: `http://www.ece.rochester.edu/~gsharma/papers/Naim_SWIFT_FCClustering_ICASSP2010.pdf`.

1.1.14 RadialSVM

Support Vector Machines (SVMs) are discrete algorithms that can be used to find the maximum margin between classes of data for the purpose of separating the data by class [21]. For a training set x, with weights w, bias b and classification values y_i on $\{-1, 1\}$ used to separate two classes n and m, SVM training is a minimization problem with the constraints:

- $x_i \cdot w + b \geq +1$ for $y_i = +1$ $(class\ n)$

- $x_i \cdot w + b \leq -1$ for $y_i = -1$ $(class\ m)$

of the function: $f(x) = sign((x \cdot w) + b)$. In concept, minimizing f with respect to this constraint finds the location of the hyperplane directly between the positive and negative training sets. The training points that lie on the hyperplanes at each edge of the margin are called support vectors. Data that cannot be directly separated by a hyperplane can be mapped to a space of higher dimensionality where separation is possible. Mapping was implemented though a non-linear transformation of the data, specifically with a Gaussian function that transformed pattern x to pattern $z = \varphi(x)$ using:

$$\varphi(x) = \frac{exp(-x^2)}{\sqrt{2\pi}}. \tag{3}$$

SVMs are commonly used tools in pattern recognition that have only occasionally been applied to flow cytometry, notably in silico [22] and on data collected from cultured cells and murine bone marrow [23]. Par of what makes SVMs appealing are that many are publicly and freely available. For this work we have used Matlab version 6.5 software (The Mathworks, Natick, MA) for pre-processing the data and then applied an SVM coded into a Matlab script by Junshui Ma and Yi Zhao of Ohio State University. SVMs require training data, and so we have entered them only into the final FlowCAP competition. The source code can be downloaded from: `http://flowjo.typepad.com/the_daily_dongle/2011/04/FlowCAP.html`

1.2 FlowCAP-II: Sample Classification Challenges

1.2.1 2DhistSVM

The key motivation for this method is that SVM can perform well in very large feature spaces. As it is possible that differences between the two groups we would like to classify may be restricted to changes in rare cell populations we reasoned that a feature extraction method that preserves much of the structure of the data at the expense of generating a lot of features may be valuable.

Feature extraction method: Our method is based on computing 2D histograms of all possible pairs of stains (forward and side scatter were ignored.). Each feature in the dataset represents the proportion of cells that fall into a specific bin on a 2 dimensional scatter plot. The bin width was set to 0.1 of the range of values in each dimension resulting in a total of 100 possible bins for each stain pair. Bins that were empty for all samples were excluded from the feature set.

Classification method: The resulting features were used for classification with SVM-struct [24] using precision-recall break-even point as the loss function. The tradeoff constant was empirically optimized over the range of 10^{-5} to 10^{3}. The number of cross-validation trials was 50 for the AML challenge and 22 for the HIV challenge (in this case this represents leave-one-out cross-validation). Classifications of unlabeled results were obtained from all cross-validation models and the results were averaged to produce final classification values. As SVM produces continuous classification values a threshold was chosen to achieve best classification accuracy on held out examples.

Implementation: Classification used SVMperfer a wrapper around SVMstruct that automates cross-validation and prediction which is available as part of the Sleipnir library (https://bitbucket.org/libsleipnir/sleipnir).

1.2.2 admire-lvq

Introduction: The first step of our analysis concerned the extraction of simple, yet meaningful features from the flow-cytometry data. In contrast to frequently employed clinical workflow, we refrained from using clustering or gating procedures to pre-select a subset of cells for further analysis. Instead, we chose to make use of all cells and all markers in a single analysis. We reduced the data to a few characteristic quantities which describe the marker statistics for each patient. Note that our feature extraction also disregards properties of individual cells or correlations between these. A particular subject was represented solely in terms of frequency counts for markers over the entire cell population. While this was meant as a first attempt, originally, the achieved excellent performance indicated that more sophisticated features were not required for the task at hand.

A data driven classifier was obtained by means of a recently developed machine learning technique: Generalized Matrix Relevance Learning Vector Quantization (GMLVQ) provides easy to implement, highly flexible, intuitive classifiers and has proven to yield competitive performance in many practical problems [25]. In contrast to many other machine learning approaches like multi-layered neural networks or support vector machines, it allows for a straightforward interpretation of the emerging classifier. GMLVQ, like other prototype-based systems, determines typical representatives of the classes which are defined in the same space as the observed data. In addition, an adaptive matrix of parameters is used to define a discriminative similarity measure. This so-called relevance matrix quantifies the discriminative power of individual features and pairs of features within the multivariate analysis. Consequently, GMLVQ provides important insights into the nature of the data set and the classification problem at hand.

Feature Extraction Method: After a first, visual inspection of histograms of the marker values for individual patients, we considered simple statistics based features derived from the data. Classification performance based on this simple choice was highly encouraging and further sophistication of the feature set did not appear necessary.

Note that the entire cell population of a given subject was considered in the feature extraction. In contrast to frequent clinical practice, no *gating* or *pre-clustering* of cells was performed. Moreover, the information of the individual cells' marker configurations was disregarded, only the frequencies of marker values in a given patient were computed and analysed.

We have not taken into account the non-specific data corresponding to tube 8. For each of the remaining 31 quantities or markers, respectively, we obtained 6 descriptive features per patient, i.e.

1. mean value
2. standard deviation
3. skewness
4. kurtosis
5. median
6. interquartile range.

For FS Lin, SS Log, and CD 45, the above were determined over all events (i.e. from 7 tubes per patient). IgG1-PE, IgG1-PC5, IgG1-PC5, and Ig-G1-PC7 were treated separately as if they were truly individual markers. Note that FS Lin was rescaled by dividing by the largest found entry to limit all observations to the interval [0,1]. For each patient the $6 \times 31 = 186$ characteristic quantities were combined into one feature vector.

Note that, in the course of the machine learning analysis described below, an additional z-score transformation was performed. From the training data only, we obtained for each of the 186 features x_j the mean value m_j over all training set patients and the corresponding standard deviation σ_j . All data (training and test) were then rescaled according to

$$x_j \leftarrow (x_j - m_j)/\sigma_j \quad \text{for} \quad j = 1, 2, ..., 186. \tag{4}$$

This z-score transformation is not essential for the performance of our algorithm, but allows for better interpretability of the resulting classifier.

Obviously, the obtained features display significant redundancy. Firstly, some of the markers may be strongly correlated, an obvious example being the above mentioned four measurements of IgG1 per patient. Moreover, one would expect a marker's mean value and median, as well as standard deviation and interquartile range, respectively, to provide highly related information. The GMLVQ approach described below, can cope with such redundancies by assigning *relevances* to features and pairs of features, thus providing an implicit, data driven weighting or selection of the most discriminative features.

Classification Method: We have employed Generalized Matrix Relevance Learning Vector Quantization (GMLVQ) in order to analyse the data and provide a classification scheme. A detailed description of the algorithm and classifier can be found in [25]. For the data set at hand, the resulting classifier is parameterized in terms of two prototype vectors (one per class) in the 186-dim. feature space, and a so-called relevance matrix Λ defining the distance measure

$$d(x,w) = (x-w)^T \Lambda (x-w) \qquad with \quad \Lambda = \Omega^T \Omega \tag{5}$$

where x is one of the 186-dim. feature vectors, w is a 186-dim. prototype vector and Ω as well as Λ are 186×186 matrices. The parameterization in terms of the unrestricted matrix Ω ensures that Λ is symmetric and positive semi-definite.

Given an LVQ system consisting of prototype w_1 (representing the class of normal patients), prototype w_2 (representing the class of AML patients) and matrix Λ, the score of an arbitrary feature vector x is computed as

$$s(x) = 2\frac{[d(x,w_1) - d(x,w_2)]}{[d(x,w_1) + d(x,w_2)]} + 1 \qquad \text{with} \qquad 0 < s(x) < 1. \tag{6}$$

Note that $s \approx 0$ indicates that $d(x,w_1) \ll d(x,w_2)$, i.e. the feature vector is much closer to the prototype representing class 1 (normal). On the contrary $s \approx 1$ corresponds to feature vectors which appear to belong to class 2 (AML). For an actual crisp classification, scores s would have to be compared with an appropriate threshold values which was, however, not requested in the challenge.

Training of the GMLVQ system

The GLMVQ classifier is obtained from the training set of labeled data $\{x_i, y_i\}$ $(i = 1, 2, \ldots 179)$, where x_i is one of the 186-dim. feature vectors and the label $y_i = 1$ if patient i is labeled as *normal*, whereas $y_i = 2$ if patient i is labeled as *AML*. Prototypes w_1 and w_2 and the matrix Ω are obtained by means of minimizing the cost function

$$E(w_1, w_2, \Omega) = \sum_{i=1}^{179} \frac{d(x_i, w_J) - d(x_i, w_K)}{d(x_i, w_J) + d(x_i, w_K)} \tag{7}$$

where the sum is over all training examples, index $J = y_i$ (i.e. the class label of the example) and $K \neq y_i$ (the wrong class). Here we refrained from introducing an additional nonlinear function, see also [25]. Numerical minimization of the cost function was done in terms of modified batch gradient descent with automated step size adaptation, obtaining, both, prototypes and the matrix Ω in the same training process.

Validation experiments

Initial experiments with the labeled data were performed in order to obtain estimates of the performance quality in terms of crisp classification. To this end, we split the available training data randomly in about 5/6 of the subjects used for training and the remaining ones as a test set. In order to obtain results independent of the precise set composition we performed averages over 50 random splits.

From the excellent performance observed in the validation runs we concluded that the resulting prediction is of high quality already in the simplest setting of the GMLVQ with one prototype per class and a single, global relevance matrix [25]. Performance also turned out very robust with respect to the number of gradient steps performed. For the final results we performed 40 steps of the gradient descent procedure. In addition we initialized the GMLVQ system randomly (prototypes close to class conditional means, matrix Ω close to identity) and averaged the resulting system over 50 initial conditions. One interesting observation concerned a particular subject: In all our GMLVQ analyses, patient 116 from the training set turned out to be very close to the typical, normal patient profile. Removing the patient from the data set resulted in close to perfect validation performances with, for instance, ROC-AUC very close to 1. We suspected that patient 116 is either an atypical case of AML or had been mislabeled in the data set. All results submitted to the challenge, however, correspond to using the full data set including patient 116.

Implementation and Availability: Feature extraction, GMLVQ training, and classification were implemented in Matlab and, consequently, are not platform dependent. All our code could be easily ported to generic programming languages.

The specific Matlab code was submitted to the challenge and the entire software is available as supplementary information: *FlowCAP-II/Attachments/admire-lvq*. Generic implementations of GMLVQ and related algorithms are also available upon request from M. Biehl (m.biehl@rug.nl).

1.2.3 biolobe

Introduction: Facing the large amount of data to be processed, a tool chain with low computational footprint was sought. Since computational methods for vector spaces are widely available, the initial task is a robust reduction of the input data of varying length to feature vectors with identical dimensions. Afterwards the correlative matrix mapping (CMM) approach is used as a subspace mapping method to project these feature vectors into a plane that allows for both visual inspection of the relationship between patients and also their analytic assessment of disease-induced mapping order.

Feature Extraction Method: Although histograms are a natural representation of marker-specific events, their appearances are dependent on the start and end of the domain binning intervals and the number of bins. Thus, in order to avoid fluctuations across adjacent histogram bins, simple density kernel estimation was carried out. A one-dimensional k-means clustering was used to determine centers. In combination with the standard deviations of their receptive fields these can be considered as Gaussian basis functions for each marker-specific event set. In order to get independence from scale and physical dimensions, the coefficients of variation (CVs) were calculated from these means and standard deviations. After all, a number of only $k = 3$ CVs per feature was chosen to provide a minimum number of degrees of freedom in the model; it turned out that larger numbers of k did not improve much the discrimination on training and validation sets for the subspace mapping method. To summarize the data pre-processing steps:

1. Blind tube 8 was always ignored, leaving 35 markers from website Table 1 plus 7×(FS and SS);

2. k-means with $k = 3$ was computed per subject and feature resulted in 49×3 dimensional feature vectors per subject;

3. in each of the three k-means-induced 1D-Voronoi cells the standard deviation was computed;

4. ratios of standard deviations and corresponding means (=coefficients of variation, CVs) were computed;

5. redundant CVs for 7 × FS Lin, 7 × SS Log and 7 × CD45-ECD were averaged;

Application of these steps resulted in $3 \times (4 \times 7 + 3) = 93$-dimensional feature vectors for each of the 359 subject.

Classification Method: For predicting the health state of patients the 93-dimensional feature vectors are mapped to a 1-dimensional subspace. To do so, a regularizing alternative to linear discriminant analysis (LDA) was applied. For the given data matrix X containing 179 training samples (rows) and its 93-dimensional feature vectors (columns) the corresponding labels L with entries of 1 for the normal state and of 2 for AML are connected by a distance metric learning framework. If D_X^V is a 179x179 V-adaptive distance matrix of the training samples, and D_L is the distance matrix of labels, being binary here, then metric parameters V are sought such that the correlation of matrix entries $r(D_X^V, D_L) = max$. The matrix elements $(D_X^V)_{ij} = \sqrt{(x^i - x^j)\, V\, V^{\mathsf{t}}\, (x^i - x^j)}$ describe the adaptive (Mahalanobis-like) matrix distance between data vectors x^i and x^j with V being optimized according to the maximum correlation mapping criterion induced by D_L. In addition to the plain algorithm described in [26] regularization is applied during optimization by clustering the mapping parameters to their k-means centers. This mapping functionality is available in the CMM software package by setting the *regul* parameter to a number between 0, excessive regularization, and 1, no regularization, i.e. describing the target fraction of the feature vector dimension.

Availability: Correlative Matrix Mapping (CMM) is online available at `https://mloss.org/software/view/293/` for use with MATLAB and GNU-Octave environments. The maximization of correlations is obtained as optimization of parameter matrix V by second-order quasi-Newton memory-limited Broyden-Fletcher-Goldfarb-Shanno (l-BFGS) optimization.

Experiment Setup: Initial CMM experiments were conducted, showing that 1-dimensional subspaces ($V = R^{93\times 1}$) are sufficient for a reliable separation of the training data. The choice of parameter regularization at regul=0.1, that is 10% of the original dimension providing 9 centers of the parameter vector (integer part of 93 × 10%), was determined in an interactive manner by looking at the separation characteristics of a 20% hold-out validation data set. By making sure that very reliable separations were obtained on random splits of the 179 training samples, confidence was created to validly utilize all these training samples in the final modeling stage. In order to deal with random initializations, 1009 (prime number for breaking ties) independent models were trained by until convergence, that is 'TolFun'=10^{-11}, 'TolX'=10^{-12}, and 'DiffMinChange'=10^{-12}. Training these 1009 models took about 3 hours on a regular desktop computer.

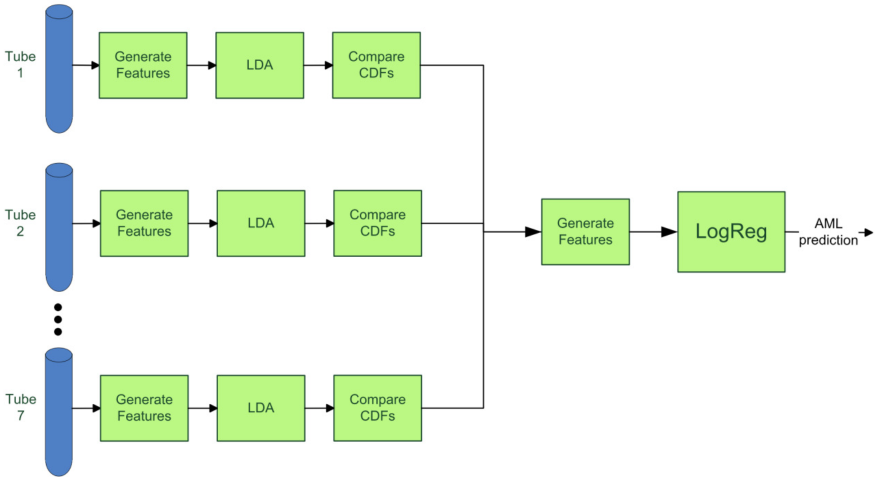
Scoring: All computed 1009 models were considered for the final scoring. If the mean of the mapped normal training samples was greater than the mean of AML, the direction of the mapping vector V was flipped to induce a reference 'normal' < 'AML' score relation for the mapping of unknown test data. The ranks of the resulting 1-dimensional projections were taken, low ranks being related to normal subjects and high ranks to AML. For summarizing the 180 ranks of test subjects across all 1009 experiments $r_{ij}, \{i = 1...180, j = 1...1009\}$ to one final value for each subject, the welfare measure of Moore was used [27]:

$$R_i = (\frac{\sum_{j=1}^{1009}(R_{ij})^{1-z}}{n})^{1/(1-z)} \tag{8}$$

with a strongly discriminative value of $z = 100$. This provides a generalization of the geometric mean used in the popular rank products method proposed by Breitling et al. [28]. Computed ranks R_i were min-max normalized to the interval $[0, 1]$ and submitted as prediction results to the competition.

1.2.4 daltons

Introduction: This solution uses simple dimension reduction and one-dimensional distribution comparison techniques to derive a set of features forwarded to a logistic regression classifier. The algorithm was one of those reaching 100 % prediction accuracy on the challenge test data. The key feature in this solution is the sparsity enforcing estimation of the classifier parameters, which discards most of the flow cytometry tubes making the model simple and economical. The following figure gives an overview of the solution.



Feature Extraction Method: Preprocessed flow cytometry data that consists of the 7 tube dependent fluorescence channels together with the forward and side scatter measurements is used. First, the dimensionality of the data from each tube is increased to 135 by appending all possible 9 inverses, 45 multiplications (including self multiplications), and 72 divisions of the measurement channels into the data. Second, the dimensionality is reduced to 1 by using Linear Discriminant Analysis (LDA). The empirical cumulative densities (EDFs) of the one-dimensional tube data are then compared against the corresponding training densities. For EDF comparison, the mean squared error (MSE) between the EDFs is used. This results in two MSE values per tube, one for AML positive and one for AML negative populations. Thus, we have a 14-dimensional vector of MSE values per patient.

Classification Method: A logistic regression model is used to map the 14-dimensional feature vector into scalar indicating the confidence of the patient being AML positive. L1 regularization of the model coefficients is used, which effectively reduces the dimension of the input vector by selecting only the relevant features and tubes. In our final submission, the L1 regularization resulted in an extremely economical model in the sense that it only used 2 of the 7 available tubes. The AML confidence score is given by the logistic regression model as

$$p = \frac{1}{1 + \exp(0.31 - 29.9\omega_4^{(0)} - 9.5\omega_5^{(0)} + 35.2\omega_4^{(1)} + 51.5\omega_5^{(1)})}, \tag{9}$$

where $\omega_t^{(c)}$ is the MSE between the tested EDF and the trained EDF of either AML negative ($c = 0$) or AML positive ($c = 1$) samples in tube $t \in \{1, \dots, 7\}$.

Below is the pseudo code for training the predictor given data `D` of all the training patients:

```
train(D) {
   for t = 1 to 7 { // For each tube
      // Generate arithmetic combinations of tube data
      X = GenerateFeatures(D(tube t))

      X0 = X(AML negative)
      X1 = X(AML positive)

      // Compute LDA coefficients
      wLda(t) = TrainLDA(X0,X1)

      // Compute EDFs in LDA projection space
      edf0(t) = EDF(MapLDA(X0; wLda(t)))
      edf1(t) = EDF(MapLDA(X1; wLda(t)))

      for p = 1 to 179 { // For each training patient
         // Compute EDF in LDA projection space
         edf = EDF(MapLDA(X(patient p); wLda(t)))

         // Calculate error with all patients' EDFs
         E(p,t,0) = MSE(edf, edf0(t))
         E(p,t,1) = MSE(edf, edf1(t))
      }
   }

   // Train regularized logistic regression model
   wLogReg = TrainLogReg(E)
   // Return trained predictor parameters
   return wLogReg, wLda, edf0, edf1
}
```

Here is the pseudo code for applying a trained predictor for one patient whose data is given in D:

```
predict(D ; wLogReg, wLda, edf0, edf1) {
   for t = 1 to 7 { // For each tube
      // Generate arithmetic combinations of tube data
      X = GenerateFeatures(D(tube t))

      // Map to LDA space
      x = MapLDA(X; wLda(t))

      // Compare EDFs
      e(t,0) = MSE(EDF(x), edf0)
      e(t,1) = MSE(EDF(x), edf1)
   }

   // Return AML confidence by logistic regression
   return MapLogReg(e; wLogReg)
}
```

Implementation: The algorithm has been implemented by using MATLAB. The program depends on a third party toolbox PMTK3 (`http://code.google.com/p/pmtk3/`).

Availability: The source code, documentation, and additional scripts for reproduction of the results are available as supplementary information in *FlowCAP-II/Attachments/daltons*.

1.2.5 DREAM–A

Introduction: In essence, we evaluated several related method by a nested cross-validation approach, and selected the best one for making predictions on the target samples. The methods are based on 2D and 3D histogram representations of each sample.

Feature Extraction Method: The histograms were generated from the preprocessed data files (*.CSV files).

2D histograms were generated for all pair-wise combinations of surface antigens and optical read-outs present on the same tube. The dynamic ranges for the readouts were assumed to be 0 to 1000 for column 1 of a *.CSV file, and 0 to 1 for all other columns. The dynamic ranges were partitioned into 50 bins, yielding a total of 2500 bins for a 2D histogram.

3D histograms were generated for all possible combinations of a subset of readouts. This subset consisted of the three readouts common to all tubes (column 1,2,5 of the *.CSV files) and the two best performing readouts from each tube. The selection of best-performing readouts was based on results obtained with predictors built from 2D histograms. For 3D histograms the dynamic ranges were partitioned into 20 bins yielding a total of 8000 bins.

A pair-wise or triple combination of readouts will be referred to as a feature. For a given feature and sample, we first count the number of cells that fall within each bin of the 2D or 3D histogram. A pseudo-count is added to each bin, and the counts are converted into fractions that sum up to one. For a given feature, we then compute a reference histogram for the classes AML and Normal, respectively, by taking the averaging over the bin frequencies over all samples belonging to the corresponding class. Note that after these processing steps the samples as well as the reference matrices are arrays of positive real numbers summing up to one; in other words, they have the properties of a probability distributions.

For each sample, we compute its distance to the two reference histograms, using one of the following distance measures: correlation distance $(1 - r)$ or conditional entropy of the sample, given the reference histogram.

Classification Method: To build a predictor, we used the distances of the training set samples to the reference histograms computed for selected features as input to a machine learning method: linear discriminant analysis (LDA), neural net, or SVM. Different configurations of neural nets and SVM were used. For each method, we selected up to five features by a recursive procedures described below. Note that an individual method is characterized by the following components:

1. A set of feature: 2D, 3D or both
2. A distance measure: correlation or entropy
3. A machine learning method (LDA, several variants of neural nets and SVM)

Each method is evaluated by the following nested cross-validation procedure. The complete training set S was split into five subsets S1 ... S5 of approximately equal size. The pseudo-code of the procedure follows:

Part 1: Computation of distance measures:

```
For all features fj :
     For k = 1 to 5:
          Make reference histogram RAk(fi) from all samples si not in Sk
           and class(si) = AML.
          Make reference histogram RNk(fj) from all samples si not in Sk
           and class(si) = Normal.
          For all si :
               DAi,k,(jj) = distance[ Histogram(si, fj) , RAk(fj) ]
               DNi,k(fj) = distance[ Histogram(si, fj) , RNk(fj) ]
          For m  k, 1  k  5 :
               Make reference histogram RAk,m(fi) from all samples si not
                in Sk   Sm and class(si) = AML.
               Make reference histogram RNk,m(fj) from all samples si not
                in Sk   Sm and class(si) = Normal.
               For all si :
                    DAi,k,,m(fj) = distance[ Histogram(si, fj) , RAk,m(fj) ]
                    DNi,k,m(fj) = distance[ Histogram(si, fj) , RNk,m(fj) ]
```

Part 2: Recursive feature selection and evaluation:

```
For k = 1 to 5 :
     Fk,0 = {} (initialization of feature collections)
For n = 1 to 5 : (5 rounds of feature selection)
      For k = 1 to 5 :
          For all features fj  not in Fk,n-1 :
               For m  k, 1  k  5 :
                    Make Predictor from all samples not  in Sk   Sm
                     using distances DAi,k,m,(fj), DNi,k,m,(fj) for all
                     features in  Fk,n-1 {fj}.
                    Use Predictor to compute Probk,n,j(si) for all si
                     in Sm.
               Compute performance measure Qk,n,j  for fi from Probk.n.j(sj)
                and class(si) using all si not in Sk .
          Select f* = best fj. according to Qk,n,j . Fk,n = { f*}  Fk,n-1.
          Make Predictor from all samples not in Sk using distances DAi
           ,k,(fj), DNi,k,(fj) for all features in  Fk,n.
          Use Predictor to compute Probn(si) for all si in  Sk .
     Compute performance measure Qn from Probn(si) and class(si) using all
      si in training set .
```

Using this evaluation procedure, the method using 2D histograms, LDA, a correlation distance measures and five rounds of features selection showed the best performance. We thus used this method to build a predictor from the entire training set and to predict the classes of the target samples.

Implementation: Evaluation and cross-validation procedures were written in R language, using the packages MASS, e1071 and nnet.

1.2.6 DREAM–B

Introduction: The following algorithm was specifically designed for the AML challenge and has its major point of innovation in the procedure for feature extraction: the main rationale beyond the procedure is to describe the wealth of measurements for each patient (31 different measures for thousands of cells) with 31 corresponding mixtures of Gaussians, each summarized by a finite number of parameters. This allowed us to capture the high-level properties of the distributions underlying each of the different measures.

For feature selection and classification, on the other hand, we exploited two recognized and consolidated techniques, namely Recursive Feature Elimination (RFE, [29,30]) and Support Vector Machines (SVM, [31]).

Feature Extraction Method: For each patient, we considered all preprocessed measurements of the 31 variables, aggregating the variables measured on multiple aliquots (Cell

Forward Scatter, Cell Side Scatter and biomarker CD45-ECD) and ignoring the noise measurements.

From visual inspection, we observed that most of the variable distributions are unimodal or bimodal, with a spike at the bottom end of the scale, possibly due to lower detection limits. We thus developed the following procedure for discriminating between unimodal and bimodal distributions:

1. For each patient and each variable, remove the bottom end spike from the variable distribution and compute maximum likelihood fits to data of both a normal distribution (10) and a weighted sum of two normal distributions (11):

$$f_{unimodal}(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} \tag{10}$$

$$f_{bimodal}(x) = p\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + (1-p)\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \tag{11}$$

2. Classify variable distributions as unimodal or bimodal exploiting the three following indicators, computed from the fit of the Equation 11:

   - The D statistic, defined as

     $$D = \frac{|\mu_1 - \mu_2|}{[(\sigma_1^2 + \sigma_2^2)/2]^{1/2}} \tag{12}$$

     where μ_1 and μ_2 are the means and σ_1 and σ_2 are the standard deviations of the two normal distributions. $D > 2$ is required for a clean separation between the two modes (as reported in [32]);
   - The misclassification area A, defined as the overlapping area under the two normal distributions (as reported in [33]). $A < 0.2$ is used as misclassification threshold;
   - The probability p from Equation 11. The minimum between p and $1-p$ has to be greater than 0.01.

   Distributions meeting all three criteria are considered as bimodal, otherwise as unimodal.

Each distribution is then described by 5 different parameters: p, μ_1, μ_2, σ_1 and σ_2 (unimodal distributions are characterized by $p = 1$ and $\mu_2 = \sigma_2 = NaN$). For the purpose of classification, we exploited parameters p, μ_1 and σ_1 of each variable distribution (further analyses including also μ_2 and σ_2 did not yield significant improvements). Each patient was thus completely described by a total of $31 \times 3 = 93$ features.

Classification Method: Classification was carried out with the Support Vector Machines classifier. To increase the robustness of the prediction, we exploited the Recursive Feature Elimination procedure. This allowed us to select a meaningful subset of the 93 features, which was then used for classification.

The classification procedure was accomplished through the following steps:

1. Choice of the kernel and tuning: we exploited a leave-one-out procedure on the training set to select the best kernel for the SVM and to tune its parameters. Linear, polynomial and Gaussian Radial Basis Function (RBF) kernels were tested. All three kernels depend on a regularization parameter C; furthermore, the RBF kernel depends on a scaling factor $2\sigma^2$ and the polynomial kernel depends on the degree of the polynomial d. Grid search with exponential step size was carried out in the parameter space, and each combination of parameters was evaluated through the Matthews Correlation Coefficient (MCC, [34]) on the samples left out by the leave-one-out procedure. MCC ranges between -1 and 1 and it is often preferred to classification accuracy when classes are highly unbalanced (as in the current case). RBF resulted the optimal kernel, with parameters $C = 2^6$ and $2\sigma^2 = 2^{10.5}$ and a MCC of 0.95.

2. Feature Selection: with a Bootstrap procedure, we obtained 100 train-test set pairs (tr_b,ts_b), $1 < b < 100$: each tr_b is obtained by sampling with replacement 179 subjects from the original train set, keeping the unsampled subjects for the corresponding test set ts_b. The Recursive Feature Elimination procedure was then applied to each pair (tr_b,ts_b): an SVM with the optimal kernel and parameters is trained on tr_b, then the importance of each feature with respect to the training error is estimated as explained in [30] and the least important feature is removed from the feature set. Then the SVM is trained again, another feature is removed and the procedure continues until only one feature remains. Every time a feature is removed, the SVM is tested on the set ts_b and the MCC of the prediction is computed. The list of removed features, read backwards, provides a ranking of the features in terms of importance for classification.

   By applying Recursive Feature Elimination to each bootstrap pair, we obtained 100 rankings of the feature set and 100 realizations of the distribution of MCC *vs.* the number of selected features. We obtained a global feature ranking by averaging the rank of each feature in the 100 feature lists. To select the optimal number of features f^* to be used for classification, we studied the curve of mean MCC *vs.* number of features: we fitted a double exponential to the curve and selected as f^* the point for which the relative increase in the fit at the subsequent point was less than 5×10^{-4}. The optimal number of features resulted 24, thus only the 24 topmost features from the global feature ranking were used for classification.

3. Re-tuning and classification: leave-one-out and grid search were used again for tuning the parameters of the RBF kernel, to optimize SVM classification for the new feature set. Optimal parameters resulted $C = 2^4$ and $2\sigma^2 = 2^{14}$, with a new MCC of 0.97. The optimal SVM was then trained on the whole train set and used for prediction on the test set. Test subjects were ranked according to real valued prediction values, scaled between 0 an 1.

Implementation: The feature estraction method is implemented in MATLAB and requires the Statistics Toolbox. Feature selection and classification are implemented with a mixture of MATLAB and Python code and require the `mlpy` Python package.

Availability: MATLAB and Python source codes of our procedure are freely downloadable from `https://www.dei.unipd.it/~dicamill/flowcap/`. The `mlpy` Python package can be downloaded from `https://mlpy.fbk.eu/`.

1.2.7 DREAM–C

Introduction:

This algorithm uses thresholding to convert signal measurements into real feature values, which are further normalized using control experiments. To avoid the problem of high dimension of feature set, Principal Component Analysis is performed at the beginning to cluster the features, and the most significant PCs are selected using cross validations. Finally, an ensemble classifier is built containing six regular machine learning methods.

Feature Extraction and Training:

Step 1: Transforming multiple measurements to a single feature value

Each CSV file contains thousands of events (rows) that correspond to the measurements for a series of cells, and 7 columns that correspond to 7 features. The first problem is how to transform these multiple measurements (rows) into one single measurement (row), which represents the values of the features for each CSV file (each sample). A naive way to summarize the multiple measurements is to use the mean. This approach, however, is inappropriate, because an AML sample likely contains both normal cells and AML cells, which means the values of each feature for this sample likely follows a bimodal distribution. Taking the mean will smooth away the measurements for AML cells in this sample. We propose to transform the multiple events for a particular feature into a single value, by calculating the proportion of the events (rows) whose value is bigger than a cutoff value K. After trying different values of K for each feature, we choose to use $K = 600$ for feature cell size (FS Lin), and K=0.6 for all other features.

For each subject, since features FS Lin and SS Log are measured 8 times from tube 1 to tube 8, and feature CD45-ECD is measured 7 times from tube 1 to tube 7, we use the median for these three features.

Finally, for each subject we have 31 features including the following: (1) FS Lin, SS Log and CD45-ECD (2) measurements of FL1, FL2, FL4 and FL5 from tube 1 to tube 7.

Step 2: Normalization against tube 8: Features FS Lin and SS Log and CD45-ECD are not normalized, and feature values from FL1 to FL5 in tubes 1-7 are normalized based on the background noise in tube 8. The calculation of normalization is very simple: subtract the feature values (defined in Step 1) in tube 1-7 by the feature values in tube 8. If after normalization, the feature value is smaller than 0, then we convert it to 0.

Step 3: Feature selection and model training: For each subject, we have 31 feature values from flow cytometry data; however, we only have 179 training samples, and especially we only have 23 AML training samples. We need to perform feature selection before model training. First, we performed principal component analysis (PCA) on all the 359 data points to generate 31 principal components (PCs) that are ranked by their % variance explained. Here is the pseudo code about how we perform feature selection on PCs:

```
1, for (i = 1; i <= 31; i++) {
2,    use the first i PCs as features
3,    evaluate model by 10-fold cross validation on the known 179 samples
4,    compute the prediction error from step 3
5,    repeat previous two steps (3 and 4) 10 times
6,    }
7, pick the smallest number of PCs that give prediction results as good as the best one
```

Classification Method:

Our ensemble classifier contains the following 6 learners: (1) linear discriminant classifier, (2) quadratic discriminant classifier, (3) support vector machine with linear kernels, (4) naive Bayes classifier, (5) logistic regression, and (6) random forests. The feature selection procedures described above suggested to use the top 3 PCs that account for about 75% of the variance within all data.

Step 4: Predicting on the test dataset: 10-fold cross validation test in Step 3 suggested us to use the top 3 PCs, hence we trained the 6 machine learning models using the top 3 PCs on all the 179 training samples, and the trained models are applied to the 180 test samples. The confidence score of classifying a sample as AML is computed as: the number of classifiers that classify a sample as AML divided by 6.

Implementation and Availability:

This program was implemented in R with some Perl scripts, and relies on some open-source machine learning packages in R. Please contact Bo Liu at boliu@umiacs.umd.edu for availability.

1.2.8 DREAM–D

Introduction: Systems biology research generates ever-increasing datasets. Unsupervised clustering is an efficient way to organize data and discover patterns, but is computationally challenging for very large datasets. As current clustering methods do not scale well, or fail to satisfy every of the important properties enumerated below, we developed a parallel computing clustering algorithm (MegaClust), which addresses all of them. We applied MegaClust to the DREAM AML flow cytometry challenge. The important properties satisfied by MegaClust are the following:

1. does not assume a specific number of clusters a priori
2. properly separate arbitrarily shapes of various densities and overlapping distributions
3. is not affected by the order in which the data is presented
4. resistant to noise (e.g. does not assign outliers to clusters)
5. uses intuitive free parameters (e.g. minimal size a population should have to be of interest).
6. is capable of completely clustering millions of observations, without data sampling or reduction.
7. parallelized (fast processing).

Methods: Our method is fully automated, does not require any human intervention, any a priori biological knowledge, any knowledge about the markers, nor any knowledge about how many patients have AML.

We used the R library PRADA to extract all the events of every patient, without any further filtering, gating, compensation or normalization, and pooled all events together. This resulted in data-sets of 10318313 to 10366310 events depending on the tube (slightly less than

359 x 30000 events, as some patients did not have 30000 cells analyzed for a given tube). We did not use the tube 8.

As we did not know a priori which surface marker were relevant for the prediction of AML vs Normal status, we decided to treat each tube independently. To further mitigate the risk of incorporating markers that were not relevant and would hence just add noise, we decided to retain only 4 out of the 5 markers at the same time. Thus for each tube, we created sub-data-sets consisting of each of the five possible combinations of taking 4 out of the 5 markers. SSC and FCS measurement were ignored.

For each sub-data-set, we performed an unsupervised clustering of the 10.4 million events using the 4 markers chosen, which were treated as a feature vector to group similar events together, based on their Euclidian distance. Only populations containing a minimum of 1000 events were retained. Depending on the sub-data- set, the number of populations identified varied between 3 and 27, plus one population containing outliers.

Outliers were discarded, and we then computed the fraction of events classified in each cell population for each of the 359 patients.

To determine which cell populations could be used to discriminate between AML and Normal, we checked if a given cell population was significantly increased in known AML patients. We retained the population for which the one-sided t.test pval was 0.1. We also checked that the size of the candidate cell population of at least one of the known AML was larger than the maximal size observed for any of the known Normal patients.

We then computed an AML score for each retained cell population. The score was defined as how many times the population size observed for a given patient was above the interquartile range of the known Normal patients.

We then checked if any cell population was significantly decreased in known AML patients, and applied the same scoring methodology.

The prediction itself was done by summing the scores obtained for every patient, and ordering them by decreasing order. We retained the list of patients with unknown status that scored above the lowest known AML patient as positive for AML. The resulting number of predicted AML patients was exactly 20, and each of these 20 patients had scores above the scores of the 3 lowest known AML. As the DREAM challenge stated that we should aim at getting 20 additional AML patients, we did not look further. However, to get a sense of the reliability of the prediction, we repeated our scoring procedure 1000 times, each time randomly leaving out 25

Implementation: MegaClust is implemented in C, and relies on the MPI libraries to runs across several nodes of the Vital-IT HPC cluster of the Swiss Institute of Bioinformatics. The extraction of measurements from fcs files and the post-analysis treatment (e.g. the leave 25

Availability: More information about the availability can be found at `http://megaclust.vital-it.ch`

1.2.9 EMMIXCYTOM and uqs

Introduction: EMMIXCYTOM (EM-based MIXture models for CYTOMetry data) is a supervised clustering method for the flow cytometry data.

To construct the distribution template of a class such as AML, EMMIXCYTOM fits a skew-t mixture model on the pool of all training samples of the class AML. Each class is acquired a template which is an object of skew-t mixture parameter list.

The similarity of a new sample to the template is measured by the approximate KL distance.

A new sample is classified into the class AML if its KL distance to the AML template is smaller than for the normal template.

Methods: There are 2872 samples in the AML data, in which the first half are labeled and the second half are to be labeled.

The first 12 AML (training) samples are used to get the AML templates, namely aml-template$_1$,..., aml-template$_7$ for each of the first seven tubes. Similarly, the first 20 normal samples are used to get the normal templates, norm-template$_1$,...,norm-template$_7$. Each template is an eight component skew-t mixture (MST) object. The 8th tube samples are treated as back-up information and are ignored in this challenge.

Then we do data analysis on each sample and assign a label to each of them. In this approach, an 8 component skew-t mixture model is fitted and the KL distances of a sample to its corresponding two tube templates are calculated and compared.

Finally,, the risk of having AML for each person is calculated based on the AML labels assigned to them, i.e the number of AML labels divided by the total successfully assigned labels for each person.

The software is available as supplementary information *FlowCAP-II/Attachments/EMMIXCYTOM*.

Implementation: EMMIXCYTOM is written in R and makes use of the EMMIX-skew library, which is written in C. EMMIXCYTOM can run on Linux, MacOs and Windows systems.

Availability: The EMMIX-skew R package is available at `http://www.maths.uq.edu.au/~gjm/mix_soft/EMMIX_R/index.html`.

Source code for the FlowCAP 2 challenges is available at: `http://www.maths.uq.edu.au/~gjm/FlowCAP/index.html`.

1.2.10 fivebyfive

We developed a supervised machine learning approach for the classification of patients with and without acute myeloid leukemia (AML). The approach uses a support vector machine (SVM) trained on feature vectors obtained from histograms of flow cytometry data for each patient. Formally, SVMs work by plotting data in a high-dimensional feature space and constructing a hyperplane that maximally separates true positives from true negatives from a set of training data.

The SVM code used for training and predicting purposes is the open-source libSVM software, available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`; the code has been modified in-house. While this software will compute the optimal hyperplane for a set of training data, there are additional inputs that must be provided by the user that can significantly affect the results. Our complete approach is based off an SVM-based algorithm we developed for the prediction of transcription factor binding sites [35, 36] and is explained in more detail there. The algorithm outline followed by detailed explanation of each line is as follows:

```
Algorithm outline

1 Map patient data to feature vectors
2 Repeat 500 times (steps 3-5):
3 Penalty parameter optimization by grid search and cross-validation
4 SVM parameter optimization using training data
5 Classify patients
6 Count runs for which each patient is classified as positive or negative
```

1. We started with the curated CSV file set. Each patient had data from 8 flow cytometry experiments. Each of these experiments had 7 columns of data (5 antibody conjugated fluorophores , plus forward scatter (FS) and side scatter (SS) measurements). For each column, we binned the data into 10 bins obtained by looking at the global maximum and minimum for each column and evenly spacing 10 bins in the interval. The value recorded for each bin was the frequency. The result is an 8 x 7 x 10 = 560 dimensional feature vector for each patient.

2. The SVM training procedure was repeated 500 times, due to the inherent stochasticity in the cross-validation procedure, discussed in step (3).

3. In the cross-validation step, the training data is randomly split into 3 sets, and each third is predicted using the SVM trained (see step (4) for details) on the other remaining two thirds. We use the F-measure, the harmonic mean of precision and recall, to assess the accuracy from cross-validation. Because of the random splitting of the data, the F-measure may vary slightly over repeated runs (typical coefficient of variance is around 0.03).

   We use the radial basis function kernel (RBF) in the SVM. There are two parameters in the RBF kernel, C and γ, which are not optimized in the SVM training procedure. Grid parameter search aims to optimize these two parameters, which are used in training the SVM: both values are varied over a coarse grid (C in $\{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and γ in $\{2^{-15}, 2^{-13}, \ldots, 2^{5}\}$) ($11 \times 11 = 121$ points in the grid), and the F-measure at each grid point is computed over 5 cross-validation repeats (to account for variations in the F-measure computation). The grid point with the largest average F is used as a starting point for a refined grid and the procedure is repeated; this refinement is carried out twice. For example, if the optimum F-measure from the coarse grid were $(C, \gamma) = (2^3, 2^{-5})$, the first refined grid would vary C over $(2^1, 2^2, 2^3, 2^4, 2^5)$ and γ over $(2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 2^{-3})$. The second refined grid would space 5 points in intervals of $2^{0.5}$ around the optimum (C, γ) from the first refined grid.

   The ultimate result of this step is the optimized parameters C and γ that are used to train the SVM on all the data in step (4). Because the optimal F-measure from this procedure depends on the random three-way splitting of the training data, the optimal C and γ change from one run to another.

4. The SVM is trained on the training data (179 patients) using the C and γ parameters from step (3).

5. The trained SVM model is used to predict the test patients as positive (diagnosed leukemia) or negative (healthy).

6. The results of step 2 (500 repeats of steps 3 - 5) are compiled. Each patient has been predicted to be healthy or have leukemia 500 times, and we determined the percentage of predictions that diagnosed each patient with leukemia.

1.2.11 flowBin

Introduction: Multiplexing flow cytometry experiments across tubes containing different combinations of markers is a common solution to the problem of measuring the expression of more markers than a particular flow cytometer can handle in one run. Data from such experiments produces unique challenges, particularly for cross centre and retrospective analysis, since markers are often assayed in different combinations. One solution is Pedreira et als method of combining tubes via K-nearest neighbours (KNN) applied across parameters shared by all tubes, to create a very high- dimensional single file. [37] However, this method implies imputation, and can lead to spurious artificial populations. [38]

Feature Extraction Method: To solve this problem, we instead binned data using overfitted K-means clustering in the shared parameters, and mapped these bins across tubes using KNN. We then extracted summary statistics (e.g., median fluorescence intensity) for each bin in terms of each parameter. Although this approach involved some data reduction, it avoided imputation.

Classification Method: Binning within patients raised the problem of linking features across patients for classification. To solve this, we took each bin from each sample as a separate training instance, labelled with the sample label, and then trained a support vector machine (SVM) classifier. For class prediction, we took the majority vote of the predicted labels for a given samples bins. Classification with parameter optimization and three-fold cross-validation was implemented using the ksvm R package, but could in theory be made to work with any modern classification method.

Implementation: flowBin was implemented in the R statistical programming language, and is dependent on the flowCore Bioconductor package as well as the class package from CRAN.

Availability: The software (in the form of an R package), documentation, and additional scripts for reproduction of the results are available as supplementary information in *FlowCAP-II/Attachments/flowBin*. We expect that a version will also be made available via Bioconductor soon.

1.2.12 flowCore-flowStats

Introduction: We applied existing BioConductor packages for flow cytometry data analysis to gate and analyze an intracellular cytokine staining assay of T-cells from HIV-vaccinated individuals, (challenge three of FlowCAP II). The goals of the challenge were two-fold: a) predict the antigen stimulation group of each sample, b) identify whether the CD4 and CD8

T-cell subpopulations for each subject were responders or non-responders to each stimulation. Our approach builds upon the hierarchical gating that is typically performed manually, however we performed data–driven automated gating rather than manual gating of each sample.

Feature Extraction Method: Using the flowStats and flowCore packages, we applied a knowledge-driven gating approach (hierarchical gating using a known hierarchy) and a new sequential normalization strategy by alternately gating and normalizing subpopulations to identify cytokine-positive, CD4 and CD8 T-cells in each sample. Normalization allowed us to use a common set of gates for each subject across stimulations, whereas cytokines were gated in a sample-specific manner to account for variation in the peak width of the cytokine-negative population. For the classification challenge, the negative control was used to compute the background adjusted proportion of cytokine positive cells for each subject. The extracted features were the proportions of cytokine positive CD4 and CD8 T–cells (i.e. the leaf nodes in the gating hierarchy) for each subject. Since the populations for each sample are defined through a common gating hierarchy, automated population matching was not required for our approach.

Classification Method: We applied a decision tree classifier (Weka), trained on the the subject–paired marginal cytokine features (i.e CD4 or CD8 T–cells marginally positive for each cytokine, paired by subject) to distinguish between antigen stimulations, under 10-fold cross validation.

Implementation: The software packages (flowCore and flowStats) used to implement our gating strategy are implemented in R and C, and are available in Bioconductor.

- flowCore: `http://bioconductor.org/packages/2.10/bioc/html/flowCore.html`
- flowStats: `http://bioconductor.org/packages/2.10/bioc/html/flowStats.html`

The R scripts for reproduction of the results are available as supplementary information in *FlowCAP-II/Attachments/flowCore-flowStats*.

1.2.13 flowPeakssvm and Kmeanssvm

Introduction: K-means algorithm was proposed more than 50 years ago and is still one of the most widely used algorithms for clustering [39]. K-means has been served very well in the data compression technique such as vector quantization widely used in signal processing. In the application of the sample classification for the FlowCAP II challenges, we used K-means with a large K (maybe much greater than the actual number of clusters) to prototype the data, the determination of the exact cluster number K is not as crucial as in the traditional clustering so we fix it to be 300. We have applied the K-means algorithm incorporating the following two techniques to achieve better separation of the K-means clusters and faster computation: i) the seeds of the K-means algorithm are generated by the kmeans++ [40]; ii) the algorithm is implemented by using k-d tree [41] to speed up computation.

The K-means algorithm is good for prototyping the data, but may not be a good choice to reveal the data clustering structure. We have developed algorithm flowPeaks [42] to address this issue. The flowPeaks combines the clusters of the K-means into a larger cluster based on the local peaks of the density function, where the density function is a Gaussian finite mixture model using the summarized data of each K-means cluster.

Feature extraction: To reduce the data complexity, we initially apply the K-means with a large K (for this challenge, K is fixed to 300) and flowPeaks to partition the cells in the training data into many clusters. The proportions of all clusters are computed and these proportions form the feature space for machine learning. **Classification method:** To build a classifier, we used the support vector machine (SVM) due to its ability to handle high dimensional data, in which many variables can be correlated. When using the proportions for the SVM input, we filter out the clusters that do not change much between the two groups either by the magnitude or by a two sample t-test. We have used the SMO algorithm [43] implemented at WEKA [44] to train the SVM machine. The performance of this machine learning technique has been assessed by the cross-validation accuracy on the training data only. The final prediction of the testing data is based on the SVM machine that is optimized by SMO by using all training data. **Implementation:** The K-means and flowPeaks are implemented initially in C++, and later packed into the R package flowPeaks. The SVM is computed using WEKA implementation.

The combination of K-means and SVM works quite well for Challenge 3 with between zero and two cross-validate errors out of 54. For Challenge 2, flowPeaks and SVM gives a slightly better cross-validation accuracy than using the plain K-means algorithm. None of the two approaches can give reasonable cross-validation accuracy for Challenge 1. In the end, we submitted K-means and SVM for Challenge 3, and flowPeaks and SVM for Challenge 2.

Specific steps for the AML challenge:

1. For each of the 8 tubes, we collect a dataset that contains all cells from “normal” patients and another dataset that contains all cells from “aml” patients.

2. For the 16 datasets (8 tubes x 2 disease conditions), we applied flowPeaks algorithm to do automatic clustering after the FS and SS channels have been removed.

3. For each original CSV file, find out the corresponding tube number, and then compute the proportions using the flowPeaks clustering from step 2. Note that two proportion vectors are computed, one for “aml” and the other for “normal”, even though the CSV file itself may be associated with just “aml” or with just “normal”.

4. For each patient (“aml” or “normal” or “NA”), concatenate all of the two proportion vectors of the 8 tubes together to form a long vector.

5. We filter out the element of the long vector (formed from step 4) where the proportion changes less than 0.1% or the p-value for the two sample t-test is greater than 0.5 between the two groups of patients. A support vector machine (SVM) with a linear kernel and scale normalization is used to build a classifier, which is used to predict the NA labeled patients. The parameters of the SVM are optimized by SMO.

Specific steps for the HVTN challenge:

1. For each CSV file that is associated with one of the two antigens (ENV-1-PTEG and GAG- 1-PTEG), cells are randomly picked and combined into a single dataset of 270K cells.

2. A K-means algorithm with K=300 is carried out on these 270K cells after the FSC-A and FSC-H and SSA channels have been removed.

3. For each original CSV file that is associated with one of the two antigens (ENV-1-PTEG and GAG-1-PTEG) or NA, we have computed the proportion of the cells that are classified to one of the 300 centers from the K-means.

4. Since each sample in the training data (antigen is labeled with ENV-1-PTEG and GAG-1- PTEG) and testing data (antigen is labeled with NA) has two antigens, the proportion has been normalized by subtracting the average proportion for the two antigens.

5. We filter out the cluster where the proportion changes less than 0.1% or the p-value for the two sample t-test is greater than 0.5 between the two antigens. A support vector machine (SVM) with a linear kernel and scale normalization is used to build a classifier, which is used to predict the NA labeled antigens. The parameters of the SVM are optimized by SMO.

Availability: The source code for the R package flowPeaks is available at `https://github.com/yongchao/flowPeaks` and at `http://www.bioconductor.org/`. The flowPeaks algorithm has been recently published in Bioinformatics [42].

1.2.14 flowType and flowType FeaLect

Introduction: Multi-dimensional cell population identification (using clustering algorithms) for exploratory analysis of FCM data is associated with several complications. First, the cell populations need to be matched to each other across multiple samples. This process has proven to be subjective, often requiring input from human experts. Second, this approach ignores the hierarchical nature the cell populations by assuming that every cell belongs to only one cell population. However, in presence of a larger number of markers, cell populations should be allowed to overlap and the computational model should be able to explore the exclusion of certain markers to determine if they are clinically relevant. Third, these algorithms cannot incorporate the background knowledge of human experts to guide the identification of rare cell populations that cannot be automatically identified. We have developed two computational pipelines for extracting a large number of overlapping immunophenotypes and for feature (immunophenotype) selection using a small training cohort. We have submitted two sets of results for challenges 1, 2, and 3:

Feature Extraction (flowType): This pipeline uses the flowMeans algorithm for cell population identification [17]. Briefly, flowMeans identifies a large number of clusters in the data and merges them based on the Mahalanobis distance between them until the desired number of clusters is reached. For each of the markers in a given dataset, flowMeans was used to identify a partition that divides the cells into a positive and a negative population. This is based on the assumption that the cells either express a given marker or not (i.e., there are two distinct cell populations). For N markers this results in 2^N phenotypes. To allow exclusion of markers from population identification (which later enabled us to identify the important markers), each marker was also allowed to be neutral (i.e., that marker was excluded from the clustering); thus, for any single subset, each marker could be negative, positive, or neutral (ignored). This increases the number of cell populations to 3^N. These phenotypes are then evaluated using ROC analysis, t-test with Bonferroni correction, and bootstrapping-based sensitivity analysis. These tests result in a hit list of ”statistically significant” features (with the exception of the HEUvsUE challenge were non of the phenotypes remained significant after p-value correction). These phenotypes are then divided to several groups, based on the Pearson correlation between them and the markers required for defining the phenotypes in each group are identified. The final representative phenotype with maximum area under the ROC on the training set is used to label the samples in the test set.

Classification (FeaLect): This pipeline builds a multivariate model using the phenotypes measures by flowType. FeaLect is specifically designed for cases in which the number of features is several orders of magnitute higher than the size of the training set. A bagging technique is used to score the features for the linear classifier. Robustness of the model is measured by both cross-validation and holdout-validation on the training-set. The model is then used to label the samples in the test-set.

Implementation: flowType and RchyOptimyx are available as a cross-platform R package and have been tested on Linux, MacOS X, and MS Windows.

Availability: More detailed descriptions of the pipelines are available elsewhere [45,46]. The R packages are available through Bioconductor and CRAN:

- flowType: `http://www.bioconductor.org/packages/devel/bioc/html/flowType.html`
- FeaLect: `http://cran.r-project.org/web/packages/FeaLect/index.html`

1.2.15 jkjg

Introduction: We solve the classification problem based on the following assumptions:

- Each experiment (tube) is an independent indication if this patient suffers from AML or not
- For each cell in a tube we may independently decide if this cell is infected or not
- The number of cells classified as infected differs substantially between patients suffering from AML and healthy patients

Feature Extraction Method: Following these assumptions we take a step-wise approach:

1. We build a classifier that returns the probability that a specific cell from a specific tube is infected or not. Such a classifier is learned for each kind of tube (that means each selection of markers) independently, where the measurements of all cells of AML patients are used as foreground (positive) and the measurements of all cells of healthy patients are used as background (negative). The log-values of the measurements are modeled by normal distributions and the parameters are learned by the maximum conditional likelihood principle.

2. For each patient, we compute the fraction of cells in a tube classified as infected. For each patient we obtain a series of 8 such fractions, one for each tube.

3. We create another classifier working on these 8 values using a logistic regression and learn its parameters by the maximum supervised posterior principle based on the labeling of patients. The output of this classifier is the final prediction.

This approach can be summarized by the following pseudo code:

```
For each tube do
    1.Load the measurements for each cell in this tube;
    2.Compute log-values for all measurements;
    3.Create a sample based on the log-values of the individual cells and label all cell
      stemming from AML patients as foreground class and all cells from healthy
      patients as background class;
    4.Create a classifier based on 7 independent normal distributions, corresponding
      to the measurements for the two scatter and five antibody measurements, for the
      foreground and background class each;
    5.Estimate the parameters of the classifier (i.e., means and standard deviations)
      from this sample using the maximum conditional likelihood (MCL) learning
      principle.
    6.Classify the log-measurements of all cells of a patient and compute the fraction
      of cells (later denoted as patient posterior) with a probability P(AML | cell) > 0.5;
Done;

For each patient do
    Create a sequence of the 8 patient posteriors;
Done;

Create a sample from these sequences with labels according to the patient's state
of health; Create a classifier based on logistic regression;

Estimate the parameters of the classifier from this sample using the maximum
supervised posterior (MSP) learning principle and a product normal prior with
standard deviation 1;
```

Classification Method:

For the predictions for the unlabeled data, we use the trained classifiers and follow the protocol as before:

- Classify each cell in each tube
- Compute the fraction of cells classified as infected
- Finally, use the classifier based on logistic regression to obtain the final prediction based on this sequence of patient posteriors.

This approach can be summarized by the following pseudo code:

```
For each tube do
    Load the measurements for each cell in this tube;
    Compute log-values for all measurements;
    Create a sample based on the log-values of the individual cells;
    Classify the log-measurements of all cells of a patient using the previously
    trained classifier for this tube and compute the fraction of
    cells (later denoted as patient posterior) with a probability P(AML | cell) > 0.5;
Done;

For each patient do
Create a sequence of the 8 patient posteriors;
Done;
```

Implementation: The approach is implemented platform independently in Java based on the Jstacs library.

Availability: Jstacs, a Java framework for statistical analysis and classification of biological sequences, can be downloaded at `http://www.jstacs.de` [Grau et al., 2012, JMLR]. The software can be downloaded at `http://www.jstacs.de/index.php/FlowCap`.

1.2.16 PBSC (Population-Based Sample Classification)

Introduction: PBSC classifies sample/subject groups based on characteristics of cell populations. Compared with approaches that do not identify cell populations but directly classify samples/subjects, PBSC provides straightforward biological interpretation. Any population identification method can be used with PBSC to form a classificatin pipeline. Based on population characteristics identified by the population identificatin method, we model the similarity between populations across different samples using F-measure-based relative distance model. Population shifts between different samples are quantified for accurate mapping. PBSC is general and can be combined with other existing automated population identification and mapping methods.

Feature Extraction Method: In PBSC, the most important feature to distinguish subject/sample groups is their population proportions. Populations are first identified and their proportions are calculated. Then T-test p-values are calculated across subject groups to identify key populations that change significantly between subject/sample groups. The key populations together with p-values are output as features for classification.

Classification Method: In PBSC, the first step of classifying a new sample/subject is identifying its cell populations and mapping them to those of the ssample/subject groups, which can be done with existing population identification and mapping methods (e.g., FLOCK used in our experiments). As the feature extraction method has identified the key populations that distinguish different sample/subject groups, a nearest neighbor classification decides which sample/subject group the new sample/subject belongs to, based on the proportions of its key populations.

Implementation: Population mapping and comparison across samples is written in C.T-test p-values are calculated within MS Excel.

Availability: Source code of PBSC can be downloaded from: `http://immportflock.sourceforge.net/`.

1.2.17 PRAMS

Introduction: Th PRAMS method uses an automated extension of the Density Based Merging (DBM) Algorithm [47] to first identify clusters, calculates each cluster's density on a per-sample basis, and then uses L1-penalized logistic regression to build a classifier for predicting sample class. It has the advantage of using pairwise projections of the data to automatically identify clusters, thus enabling each automatically identified population to be easily re-identified by manual gating.

Method Description: All sample data is first combined into a single aggregate data file. Next, clusters of cells are identified by first projecting data in all pairwise sets of dimensions and then using DBM to identify single pair of dimensions that contains the highest confidence cluster. Cells from each identified cluster are again then projected into all pairwise dimensions to identify the next most informative set of dimensions. This process is recursively repeated until no new clusters are found. The number of cells in each cluster is then computed on a per-sample basis and these features are used to train an L1-penalized logistic regression classifier. Cross validation is used to select the optimal regularization constraint which is used to constrain the final classification model. This model was then used to predict the labels of each test-set sample.

Implementation: Clustering is performed using a Matlab-implementation of DBM. All feature extraction and classification is performed in R.

Software Availability: PRAMS software is available on the Nolan Lab Github website: `https://github.com/nolanlab/flowcap2`

1.2.18 Pram Spheres, CIHC, & Random Spheres

Introduction: Our approach was focused at providing good classification, thus we have effectively dropped the gating step. Instead of looking for cell populations first and then at the event density differences inside them, we have been directly looking for regions that are useful for classification. Precisely, we have been automatically creating large amounts of small gates covering interesting regions and combining their predictions using various ensemble schemes to provide accurate and non-overfitted classifiers. To this end we have employed the perambulating spheres algorithm which is using genetic algorithm to build those gates and the gradient boosting scheme to assemble them into a final model.

Feature Extraction Method: The core of our approach is the idea of describing the density of events on the descriptor space by its fuzzy overlap with a N-dimensional spherical Gaussian density distribution described with its centre $\vec{\xi}$ and radius r. The overlap is given by

$$k(E,\vec{\xi},r) := \frac{1}{\sum_{\vec{x}\in E} 1} \sum_{\vec{x}\in E} e^{\frac{||\vec{x}-\vec{\xi}||^2}{r^2}},$$

where E is a set of events $\vec{x}_i$. Thus, such sphere alone could be perceived as a simple learner which, for some event set, returns a score the higher the bigger probability it is that this event set has a certain decision. We have used a genetic algorithm to find spheres for which the values of overlap correlate best with the decision. Due to our simultaneous participation in DREAM5, for the challenge 2 we have employed an earlier, simpler variant of this method, Random Spheres, in which the overlap between the spheres and the event density was treated in a crisp way, i.e. without gaussian membership weight:

$$k(E,\vec{\xi},r) := \frac{1}{\sum_{\vec{x}\in E} 1} \sum_{\vec{x}\in E} \begin{cases} 1 & \text{for } ||\vec{x}-\vec{\xi}|| < r \\ 0 & \text{otherwise} \end{cases}.$$

Classification Method: Unfortunately the accuracy of the optimised sphere is not satisfying and the process is prone to overfitting. Thus, we employ a simple stochastic gradient boosting with shrinkage [48] to combine several spheres build on a different subsets of the training set and on residuals of a previous sphere's predictions as a decision. To be able to reasonably combine predictions of spheres within ensemble, we have been using simple, OLS linear model to rescale the spheres' predictions to the range of current pseudoresiduals.

Implementation: Our method is implemented as an R package and does not depend on any other software.

This is the pseudocode of the full algorithm. N denotes the dimensionality of the descriptor space, N_s number of samples, N_g number of genetic algorithm generations, σ is the shrinkage parameter and δ is the depth of boosts.

Prediction with a single sphere:

```
$\leftarrow E_i$ {event set for some sample $i$}
$\leftarrow \vec{\xi}, r$ {sphere}
for $i \leftarrow 1$ to $N_s$ do
    $d_i \leftarrow k(E_i, \vec{\xi}, r)$
end for
return $d_i$
```

Building a sphere:

```
$\leftarrow E_i$ {event set for each sample $i$}
$\leftarrow d_i$ {decision for each sample $i$}
$\mathcal{E} \leftarrow \bigcup_i E_i$
{Initial population}
for $j \leftarrow 1$ to 100 do
    $\vec{\xi_j} \leftarrow$ random element of $\mathcal{E}$
    $r_j \leftarrow$ random number from $U(0.05, 0.5)$
end for
{Evolution}
for $k \leftarrow 1$ to $N_g$ do
    for $j \leftarrow 1$ to 100 do
        $S_j \leftarrow$Correlation(predict_with_sphere($E_i, \vec{\xi_j}, r_j$), $d_i$)
    end for
    Sort $\vec{\xi_j}$ and $r_j$ in order of decreasing $S_j$
    {Leave the best sphere untouched, ...}
    $j \leftarrow 2$
    {... mutate top-20 with certain repetitions, ...}
    $\vec{\xi_j^*} \leftarrow \vec{\xi_j}$
    for $l$ in $\{1-10, 1-10, 1-9, 1-20\}$ do
        $\vec{\xi_j} \leftarrow \vec{\xi_l^*}+$ vector of $N$ random numbers from $N(0, 0.05)$
        if random number from $U(0,1) > 0.7$ then
            $r_j \leftarrow$random number from $U(0.05, 0.5)$
        end if
        $j \leftarrow j + 1$
    end for
    {... finally replace the rest of spheres with new random ones}
    for $j \leftarrow j$ to 100 do
        $\xi_j \leftarrow$ random element of $\mathcal{E}$
        $r_j \leftarrow$ random number from $U(0.05, 0.5)$
    end for
end for
return $\vec{\xi}_1, r_1$
  Training:
$\leftarrow E_i$ {event sets for each sample $i$}
$\leftarrow d_i$ {decision for each sample $i$}
for $j \leftarrow 1$ to $\delta$ do
    $\vec{b} \leftarrow$ random subsample of $\frac{2}{3}$ of elements of $\{1, \ldots, N_s\}$, with the same proportion of classes
    $\vec{\xi}_j, r_j \leftarrow$ build_sphere($E_{\vec{b}}, d_{\vec{b}}$)
    $a_j, b_j \leftarrow$ coefficients of a simple linear model fitted to $d_{\vec{b}}$ ~predict_with_sphere($E_{\vec{b}}, \vec{\xi}_j, r_j$)
    $d_i \leftarrow d_i - (1 - \sigma) \times (a_j \times$predict_with_sphere$(E_i, \vec{\xi}_j, r_j) + b_j)$
end for
return $\vec{\xi}_j$, $r_j$, $a_j$, $b_j$
  Prediction:
$\leftarrow E_i$ {event sets for each sample $i$}
$\leftarrow \vec{\xi}_j$, $r_j$, $a_j$, $b_j$ {boosting ensemble of spheres}
$d_i \leftarrow 0$
for $j \leftarrow 1$ to $\delta$ do
    $d_i \leftarrow d_i + (1 - \sigma) \times (a_j \times$predict_with_sphere$(E_i, \vec{\xi}_j, r_j) + b_j)$
end for
return $d_i$
```

The selection of the hyperparameters of this algorithm (N_g, σ and δ) depends on the problem; the reasonable default values are $N_g = 20$, $\sigma = 0.5$ and $\delta = 10$.

The correlation function is a simple Pearson correlation coefficient when we want to find some specific state among 'normal' samples (as in challenge 1 and 2) and an absolute value of it when we want to distinguish two different states (as in challenge 3a).

As usual with stochastic learners, the performance of a single sphere boost is quite variable. To cope with that, and also further increase the accuracy, we have imposed yet another level of ensemble learning. Namely, we have been building the learners in a 64-fold bagging loop, i.e. on M random subsamples of the training set, each containing $\frac{2}{3}$s of samples selected in a way to retain the class proportions.

In all challenges we have calculated a simple median of all predictions from all bagging iterations and used it as the final score. While in both challenges we knew how many samples in the test set should have which class, we have just ordered the samples in order of their scores and label them accordingly to satisfy the count constrain.

Availability: Our tool is available from the following URL: `http://bioinfo.icm.edu.pl/pramSpheres.tar.gz`.

1.2.19 BCB and SPADE

Introduction: Flow cytometry and the next-generation mass cytometry technologies capture the heterogeneity of biological systems by providing multiparametric measurements of single cells. Even as cytometry technology is rapidly advancing, methods for analyzing this complex data lag behind. Especially that the advent of mass cytometry (CyTOF) is quickly increasing the dimensionality of the data to 30+, making the traditional analysis approaches a critical bottleneck. To objectively explore the richness of such high-dimensional single-cell data, new computational methods are needed.

We developed a novel analytical approach, Spanning-tree Progression Analysis of Density- normalized Events (SPADE), to explore high-dimensional cytometry data in a robust and unsupervised manner, and reveal a likely underlying cellular hierarchy [Qiu, et al, *Nature Biotechnology*, 2011]. Briefly, SPADE views a cytometry dataset as a high-dimensional point cloud of cells, and uses topological methods to reveal the geometry of the cloud. To solve the challenges in FlowCAP2, we coupled SPADE with the Earth Mover's Distance (EMD) and the Relief classifier, which is a nearest neighbor classification algorithm.

Feature Extraction Method used for AML prediction:

1. **Initial inspection of the CSV data:** Look at the mean and variance of each channel across all files, and determine whether normalization is needed. The conclusion was: the FSC channel needs to be normalized to mean=0, std=0.1; the other channels do not need normalization.

2. **Divide the data into 7 sets, and analyze them separately:** Since there are 7 staining panels with minimal overlapping, a joint analysis is probably not the best idea. Therefore, the data of all 359 samples were divided into 7 subsets. Each subset contained the CSV files for 359 samples, one staining panel. The 8th panel (iso) was not used for subsequent analysis.

3. **SPADE analysis of each of the 7 subsets separately:** SPADE stand for Spanning-tree Progression Analysis of Density-normalized Events. This is a new approach for analyzing flow cytometry data, and is published in Nature Biotechnology in 2011. Briefly, SPADE views a flow cytometry dataset as a cloud of cells, and derives a tree structure to represent the shape of the cloud. Each node can be viewed as a mini-gate. The tree connecting the gates may reveal the underlying cellular hierarchy. For one of the 7 subsets, we perform the following: (1) downsample the FCS file of each sample, reduce the number of cells; (2) pooled the data of all 359 samples together, obtaining a big FCS file that represents the union of all samples; (3) perform agglomerative clustering; (4) construct a minimum spanning tree to represent the linkage among the clusters; (5) for each of the 359 samples, calculate the percentage of cells in each cluster.

4. **Extracted feature**: After the above steps, we obtained 359 distributions (how cells in each sample distribute across the tree). This cellular distribution can be viewed as a characteristic of each sample.

Classification Method used for AML prediction:

1. **Classification using EMD and NN:** We use the cellular distributions, the tree structure, and the Earth Mover Distance (EMD) to define distances / dissimilarities between each pair of samples. For each testing sample, we calculate the difference between its distance to its nearest normal sample and its distance to its nearest normal sample. The difference between the above two distances is compared with 0, in order to make a decision whether this testing sample is normal or aml.

2. **Result summary:** We performed 3 and 4 for each of the seven datasets (corresponding to the seven signal tubes). Therefore, we have seven classifiers, and a total of seven sets of prediction results, one from each classifier. It turned out that the prediction results of the seven classifiers are highly consistent, meaning that the seven classifiers gave similar answers. Their consensus was the basis of our final prediction.

Method used for the HVTN stimulation prediction:

1. **Initial inspection of the raw fcs:** Look at the mean and variance of each channel across all files, and determine whether normalization is needed. We chose to use hyperbolic inverse sine transformation, with co-factor = 100.

2. **SPADE analysis of 96 files jointly:** For this challenge, we jointly analyzed the data of 96 files [(24 training + 24 testing) * 2 stimulations]. We performed the following: (1) downsample the FCS file of each of the 96 samples, reduce the number of cells; (2) pool the data of all 96 samples together, obtain a big FCS file that represents the union of all samples; (3) perform agglomerative clustering; (4) construct a minimum spanning tree to represent the linkage among the clusters; (5) for each of the 96 samples, calculate the percentage of cells in each cluster. After the above steps, we obtained 96 distributions (how cells in each sample distribute across the tree). This cellular distribution can be viewed as a characteristic of each sample.

3. **Classification using PCA:** For each of the 96 sample, we subtract its cellular distribution by its paired sample (same sample has two stims). We observed that (ENV-1-PTEG minus GAG-1-PTEG) and (GAG-1-PTEG minus ENV-1-PTEG) of the training samples formed two distinct patterns in PCA space. This was the basis for our classification analysis.

Implementation: SPADE was originally developed and implemented in Matlab. An enhance versiond, SPADE2 was used to perform the analysis described above. Compared with the original version, SPADE2 is 20 times faster and has interactive graphical user interface. SPADE has also been re-implemented efficiently in R and Cytoscape.

Availability: Details of the SPADE algorithm is described in Qiu, et al, *Nature Biotechnology*, 2011. The Matlab source-code of SPADE is available at *FlowCAP-II/Attachments/SPADE*. The Matlab-based implementation, SPADE2, is available at `http://odin.mdacc.tmc.edu/~pqiu/software/SPADE2/index.html`. The R/Cytoscape version of SPADE is named CytoSPADE and is available at `http://cytospade.org/`.

1.2.20 SPCA+GLM

Introduction: The given data pertain to study of an immunological disease, namely leukemia. Measurements from single cell level were taken for 179 individuals of which 23 were AML patients. Flow-cytometry technique was used to study the samples. Further description of the original data can be found above.

We have implemented a combination of supervised principal component analysis (*SPCA*) and generalized linear model (*GLM*) as framework for building the predictive model. Since the objective is to be able to predict successfully the disease status of new sample, model assessment was carried out based on 4-fold cross validation. While use of *SPCA* and/or *GLM* is not novel however in numerous occasions we have noticed inappropriate or careless use of these techniques for building predictive model. Thus in every level of model building and prediction caution was taken to ensure stability of the learning samples across folds, we ensured against leakage of information from test set, numerical accuracy as much as computationally feasible , etc.

Feature Extraction Method: Furthermore the challenge in this particular data was to derive useful predictor variables to be used for model building. It was noticed that a varied number of observations were measured for each subject from each tube with minimum of 9252 to a maximum of 29920 observations for any single subject from any *tube*. The average number of observations for any individual did not seem to be affected by the disease status. Thus, instead of depending on actual number of observations from an individual, percent frequencies of different attributes were used. Note that seven types of features from each of the eight *tubes* together yielded 56 types of measurements for any single individual. Using the sample percentiles (over all individuals) as bins for each feature we derived estimate of marginal distributions of these attributes and computed percent frequency of observations for each bin (for each feature of each subject). Although initially 11 percentiles for the 56 features were used, some appeared redundant and altogether 519 such bin-based frequencies were used as (summary) covariate for each individual.

Classification Method: Feature selection: Noting that we have data on 179 individuals and in fact for the 4-fold validation described above we would have approximately only 135 subjects as learning set within a fold, it was essential to carry our variable selection. As mentioned earlier supervised principal component method would be used for this purpose. For the supervised part noting that response is binary while predictors are continuous we chose the following measure to ascertain usefulness of a predictor for the response variable (i.e. disease status).

For each fold we computed the means for normal and AML subject for each of the 519 covariates. Higher difference in the absolute mean difference would indicate usefulness of a covariate to distinguish between the normal and AML subjects. In subsequent discussion we would refer to these measurements as abs-MDF (absolute mean difference). Note that abs-MDFs were calculated for each fold-specific learning set. Different (percentile based) cut-offs on the 519 abs-MDF values were used select top covariates into the model. Thus different learning sets could very well use different sets of top covariates for same cut-off value.

Further dimension reduction was by Principal component analyses on the selected covariates for each such cut-off on the abs-MDF measurements. Once again PCAs were carried out for different fold specific learning sets. The corresponding rotation matrix was used to the remaining subjects for that fold which are used to assess out-of sample prediction quality.

Classification: The principal components thus obtained were used as regressors into logistic regression models. For each cut-off on abs-MDF varied numbers of principal components were used in the regression model to determine the optimal choice of cut-off and optimal number of principal components to be used to generate best predictive results. Altogether 22 different choices of cut-offs on abs-MDF were used and for each cutoff 35 different choices of (number of top) principal components were considered, giving rise to 770 possible model choices attempted.

In this exercise it appears several choices provide comparable results. We chose top 40% covariates of the 519 covariates. As mentioned before the set of these top covariates would be fold-specific. For the corresponding covariate matrix first 55 principal components were then used in the logistic model.

Implementation and availability: The code for prediction is available in *R* and does not require any specific libraries and is available upon request. However the method of feature extraction was carried out partly in *R* and partly in it Excel (manually) thus as yet not readily available to be implemented elsewhere.

1.2.21 SWIFT

Introduction: Variability in clustering poses a challenge for further analysis of samples based on features derived from clustering. To effectively exploit SWIFT in the supervised learning tasks involved in FlowCap II, a SWIFT-based (see FlowCAP 1 for a description) co-clustering methodology was adopted to overcome the hurdle posed by variability in clustering.

Method: SWIFT (see FlowCAP-I for a description) was used to cluster a concatenate of training samples. The output mixture model representation provided by SWIFT was then used to perform assignment of events in training and test samples to clusters. The cluster membership counts, i.e. the numbers of events per cluster in the sample, were then utilized as features in a standard supervised classification framework utilizing a support vector machine (SVM). The SVM was trained using the known sample labels for the training samples and the resulting trained SVM was used to classify the test samples, in both cases using the cluster assignment counts as features.

Implementation: SWIFT is implemented in Matlab and uses the Matlab Statistics Toolbox. The SVM based classifier was also implemented using the SVM toolbox in Matlab.

Availability: See FlowCAP 1 description of SWIFT.

1.2.22 team21

Introduction: The approach uses relative entropies to evaluate if the distribution of the values of flow cytometry data for a given individual is closer to the overall distribution for AML or for Normal individuals in the space of values Γ.

Feature Extraction Method: Entropies are computed for Γ=(”FS Log”, ”SS Log”, ”FL3 Log”, j) with $j \in$ {”FL1 Log”,”FL2 Log”, ”FL4 Log”,”FL5 Log”} for Tube $k \in \{1,2,3,4,5,6,7\}$. The total relative entropy difference for individual i, $\Delta S_i = \sum_{j,k} \Delta S_{i,j,k}$, is defined as the sum of all the contributions.

Classification Method: The relative entropy with respect to AML minus that with respect to Normal individuals indicates that the individual looks like an AML patient for positive values and like a Normal subject for negative values.

Implementation: The algorithm is implemented in python using standard libraries.

Availability: The details of the approach, the source code, and the instructions to execute the code are available at: `http://www.ehu.es/biologiacomputacional/team21_vilar`

2 Supplementary Note 2: Refined Manual Gates

2.1 Problems with Manual Gating as a Gold Standard

A surprising aspect of the FlowCAP results is the variability that individual algorithms exhibited across challenges in some cases. As an example, the flowClust/flowMerge algorithm performed poorly in Challenges 1 and 2 across nearly all data sets, while performing quite well in Challenge 4, even though the participants did not train their algorithm for Challenge 4. For Challenges 1 and 2, flowClust/flowMerge was applied sequentially, first to the scatter dimensions, and then to the fluorescence dimensions for each population identified in the scatter channels. However, for Challenge 4 the FSC/SSC dimensions were excluded from prediction algorithm, since it was clear from the training data that they were not used for manual gating. Consequently, the difference in F-measure for flowClust/flowMerge between Challenges 1 and 4 was due to the inclusion or exclusion of the scatter channels in automated gating.

Closer examination of the data showed that the FSC/SSC channels contained structure that was not captured by the manual gates (data not shown). When we examined the output of the flowClust/flowMerge algorithm, we found that 1: it captured the structure of the data in FSS/SSC, and 2: it found sub-clusters within each FSC/SSC population based on the fluorescence markers (data not shown). To further investigate this, we computed the F-measure for the flowClust/flowMerge algorithm conditional on the FSC/SSC subpopulation. That is to say, for each FSC/SSC subpopulation, we computed the F-measure with respect to the same subset of cells using the manual gate assignments. We repeated this analysis across all samples and data sets for which manual gates were available in the original FlowCAP data (15 samples in all, 3 per data set). We found that conditioning on the FSC/SSC populations significantly improved the F-measure for the algorithm (paired t-test, $\Delta\mu = 0.167$ p $= 9.5 \times 10^{-5}$ vs manual gating and $\Delta\mu = 0.152$ p $= 2.1 \times 10^{-5}$ vs flowMeans.

These results indicate that flowMeans, manual gating, and flowClust/Merge agree extremely well on some subpopulations of cells defined in the FSC/SSC dimensions. However, incomplete manual gating combined with a global cluster agreement statistic penalized the flowClust/flowMerge F-measure due to increased cluster heterogeneity in the manually defined (but arguably erroneous) clusters. Note that this effect is likely not restricted to the flowClust/flowMerge algorithm, but the two-stage gating approach undertaken in Challenges 1 and 2 by the flowClust/flowMerge authors probably exacerbated this anomaly.

These findings highlight the problem of using unsupervised manual gating as the reference for F-measure determination. In addition to problems associated with the subjective, non-deterministic nature of manual gating, the subject matter experts who provided the manual gating results did not attempt to identify all cell populations apparent in their datasets (exhaustive gating). As is often the case, experimentalist often focus on a subset of cells based on their underlying hypothesis or physiological process of interest, essentially ignoring other structures in their data. However, the automated algorithms were design to provide a complete dissection of the population structure in a given dataset. Therefore, in some cases relatively low F-measure values are not an indication of poor algorithm performance, but rather problems associated with incomplete manual gating. We tried to further investigate this issue by analyzing additional manual gates produced by eight independent experts who were instructured to identify all cell populations discernible from the available data (exhaustive gating).

2.2 Additional Manual Gates

2.2.1 Instructions for Manual Gaters

These instructions explain how we guided manual gating by multiple operators to achieve ”good current practice” gating, without providing too much guidance that might make the manual gates artificially consistent.

Technical notes:

There are 30 fcs files located in the StemCell directory, and 12 fcs files located in the GvHD directory. FlowJo version 9.3-9.4 or higher is recommended for analysis. Adjust Flowjo settings so that all parameters FSC, SSC and the fluorescence parameters display as linear (the data has already been transformed).

Strategy:

To compare automated algorithms with the current practice of manual gating, we established a set of guidelines that were given to eight experienced flow cytometrists, who then constructed manual gates for all discernible populations (a.k.a. exhaustive gating) without further discussion between them. The design of this manual gating process is intended to produce good-quality and reasonably consistent manual gates by minimizing some sources of variation such as 1) including all or almost all data events and 2) avoiding overlapping gates. Subjective decisions regarding the number of populations and the cutoffs between populations were left to the individual operators to reflect current practice.

Guidelines:

1. If samples have been pre-gated for FSC and SSC, i.e. the Hematopoietic Stem Cell Transplant data (HSCT), these parameters do not need to be used for further gating. In the case of the GvHD data, draw as many gates as needed to logically define distinct populations in the FSC and SSC dimensions. These gates can be applied to all samples if desired. Some individual gate adjustment may be needed in some samples to achieve clean population separation. It is possible that not all populations are present in all the samples.

2. Using either the whole population (HSCT) or each FSC vs SSC-defined population (GvHD), examine 2D plots of all possible pairs of parameters (FL1vs2, 1vs3, 1vs4, 2vs3, 2vs4, 3vs4) and choose one plot for the further gating. Use whatever gating strategy that allows you to clearly capture every distinct cell population observed in that 2D plot. The end goal is to separate all populations that exist in this 2D plot, being as inclusive as possible so that few if any cells are unassigned to any population. No event should be included in more than one of the gated populations. This process should be repeated for each of the FSC vs SSC populations.

   Decision required: For some samples the values of some parameters may be artificially assigned to the first data bin and may appear along the axis of the plot. These points may appear to be part of a negative population, i.e. continuous with the ’low’ cells, or they may appear to represent a separate population. You should use your best judgment to decide whether to gate on these zero value events as distinct populations or not.

   Decision required: Ellipsoid, rectangular, quadrant, spider or polygon gates may be used, as appropriate. Quadrant and spider gates have the advantage that all cells are included, whereas polygons provide the greatest opportunity to match the gate shape to the population. If quadrants or spiders are used, include only those quadrants that contain significant populations, i.e. not all possible populations may exist.

3. Plot each gated population from step 2 on the other two fluorescent parameters, and draw more gates to encompass all populations that appear to be distinct. The end goal is to separate all populations that exist in the data, being as inclusive as possible so that few if any cells are unassigned to any population. No event should be included in more than one of the final gated populations.

4. When you are satisfied with your results, save your workspace as an xml file (File Save as XML) and send to Nima Aghaeepour at naghaeep@bccrc.ca. Please use the 9.x mac version of flowjo and save your workspaces as version 2 XMLs. Come up with a consistent format for labeling your gates and provide an excel spreadsheet of the labels that correspond to the gates that should be included in the analysis (i.e., the final cell populations). For questions regarding the data format or if you dont have access to flowjo, please contact FlowCAPs organizing committee.

3 Supplementary Note 3: Independent Analysis of the AML Dataset by the DREAM Initiative

The participants to the DREAM6/FlowCAP-II challenge were required to submit a list of subjects ordered according to the confidence assigned to the subject being affected with AML. That allowed us to compute more metrics than the ones used in the other FlowCAP challenges and assign an order to the teams that had perfect classification. We received 17 predictions, 8 of them achieved perfect classification. The lowest precision and recall were 0.80. We scored participants using the average of 4 metrics: Precision, Recall, Matthews Correlation Coefficient and Jaccard Similarity Coefficient (Supplementary Table 2). Given that 8 teams obtained perfect score, we decided to create an ad-hoc score given by the Pearson correlation between the confidence levels provided in the submission and the “perfect” confidence level (1 for AML patients and 0 for healthy donors) to further rank those 8 teams. This new metric created some interesting discussion, as archived in the discussion group (`http://www.the-dream-project.org/forum/29`). This discussion reflects the fact that there is always some arbitrariness in the choice of performance metrics.

A team’s rank is determined by the average of the following 4 metrics, which take into account that 20 out of 180 subjects were AML positive.

- Prec: The Precision of the predictions, defined as the fraction of correct AML patients amongst the first 20 predictions. Precision is a measure of accuracy.

- **Rec**: The Recall of the predictions, defined as the proportion of AML patients in the first 20 predictions out of all the AML patients in the cohort. Recall is a measure of completeness.

- **MCC**: The Matthews Correlation Coefficient is a measure of the quality of binary classifications. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. MCC is similar to the Pearson correlation coefficient in its interpretation. $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}}$

- **JSC**: The Jaccard Similarity Coefficient (also known as the Tanimoto coefficient) measures similarity between two sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. In this case, set 1 is composed by the first 20 subjects predicted to be affected by AML, and set 2 is composed of all the 20 AML patients.

4 Supplementary Note 4: Post-hoc Analysis of the HVTN Dataset

A post-hoc analysis of the ICS data was performed to compare the features selected by manual gating against those selected by some of the automated gating methods. Specifically, one of the algorithms `flowCore-flowStats`, was designed to mimic the manual gating hierarchy for this data set (the positions of the gates were derived automatically from statistical analysis of the data). This pipeline allowed us to make a direct comparison of the features identified by automated and manual gating.

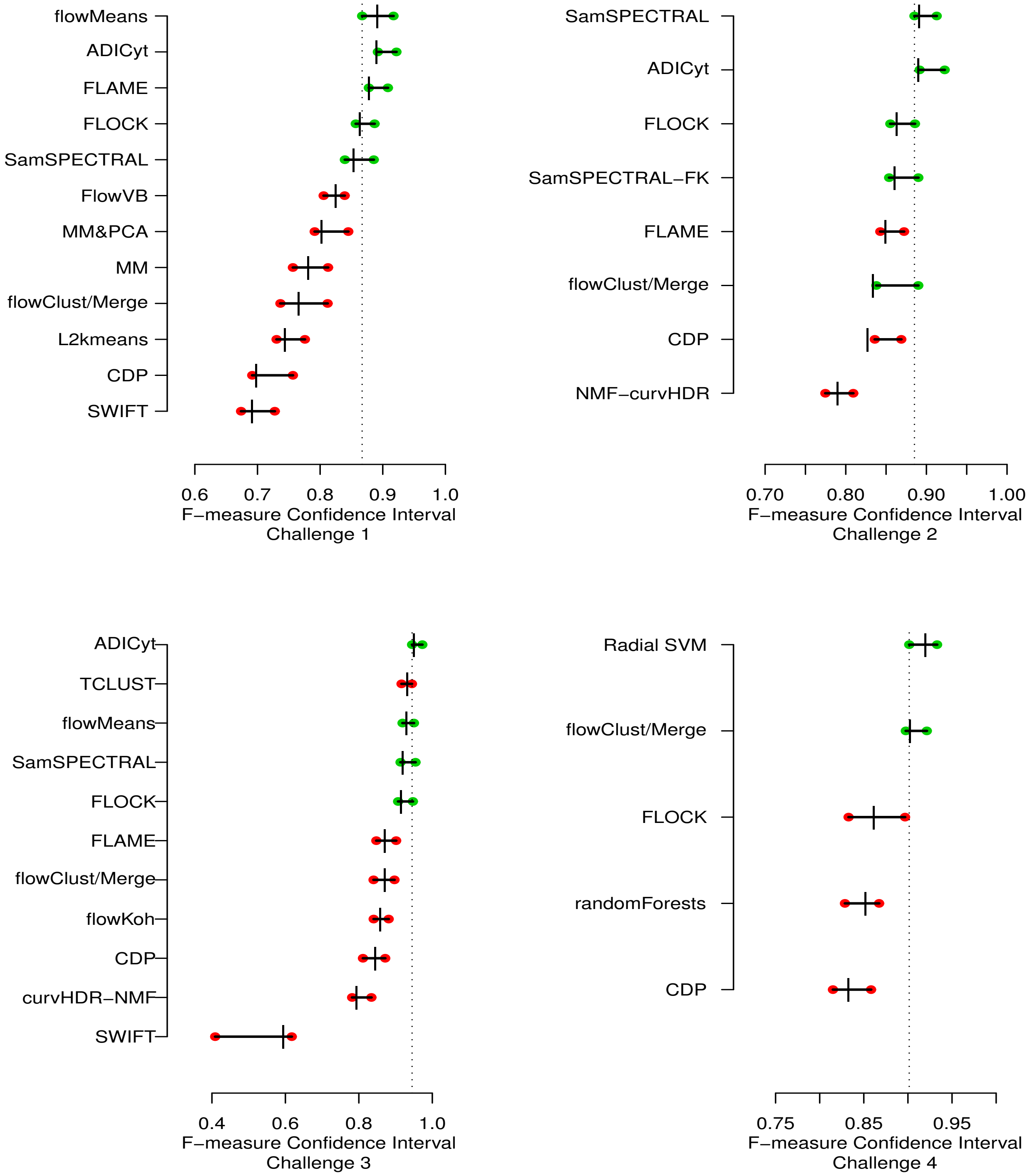
For the features extracted from the automated gates, we compared the paired proportions of cytokine positive cells in the ENV and GAG stimulated samples and found that the ENV stimulated samples had systematically higher IL2+/CD4+ proportions than the same sample stimulated with Gag. The rule for predicting Env vs Gag was to classify the sample with the higher proportion of IL2+/CD4+ cells as Env, and classify the other paired sample as Gag. In the training and testing data, this rule gave perfect classification. Supplementary Figure 20 (top middle panel) shows the difference between paired Env and Gag stimulated samples for various cytokine combinations. CD4+/IL2+ is the only combination that discriminates perfectly between Env and Gag stimulations (Env is always larger). When these same features were examined in the manually gated data (Supplementary Figure 20 top left panel), we found that the Env vs Gag paired difference for the CD4+/IL2+ population did not discriminate perfectly between stimulations but there was a trend towards an increased IL2 response in the Env stimulated samples. In order to validate whether this effect was caused by vaccination, we examined the placebo–treated group of samples from the same trial and performed the same automated and manual gating analysis of that data. In the placebo–treated group, the Env vs Gag paired difference of IL2+/CD4+ cell proportions did not discriminate between stimulations for either automated or manual gating, suggesting that the effect was specific to vaccination (Supplementary Figure 20 top right and left panels). Interestingly, closer post-hoc examination of the data revealed that several of the control and stimulated samples in the data set were matched from different experimental runs (i.e. control and stimulation samples were taken from different plates), suggesting a possible run–specific effect. When these samples were filtered out of the analysis, manual gating was also able to perfectly discriminate between Env and Gag stimulated samples in the vaccinee group based on the CD4+/IL2+ population (Supplementary Figure 20 bottom row).

Other algorithms, though not directly comparable to manual gates, also identified features with the IL2+/CD4+ phenotype that could discriminate between Env and Gag stimulation. These results demonstrate that automated gating can perform as well as manual gating, even exceed the performance of manual gating in some cases, as evidenced by the ability of automated gating to discriminate between Env and Gag stimulated samples even in the presence of specific technical bias. Automated gating can reduce gating variability and increase statistical power to detect differences between samples that might otherwise be missed by manual analysis. This is very important in vaccine trials and ICS assays where effects of interest are often very small. The CD4+ specific effect observed for the Env stimulation in the HVTN049 data is explained by the protein boost given in the form of gp140 protein (a subunit of Env), thus eliciting an MHCII response through CD4 T–cells.

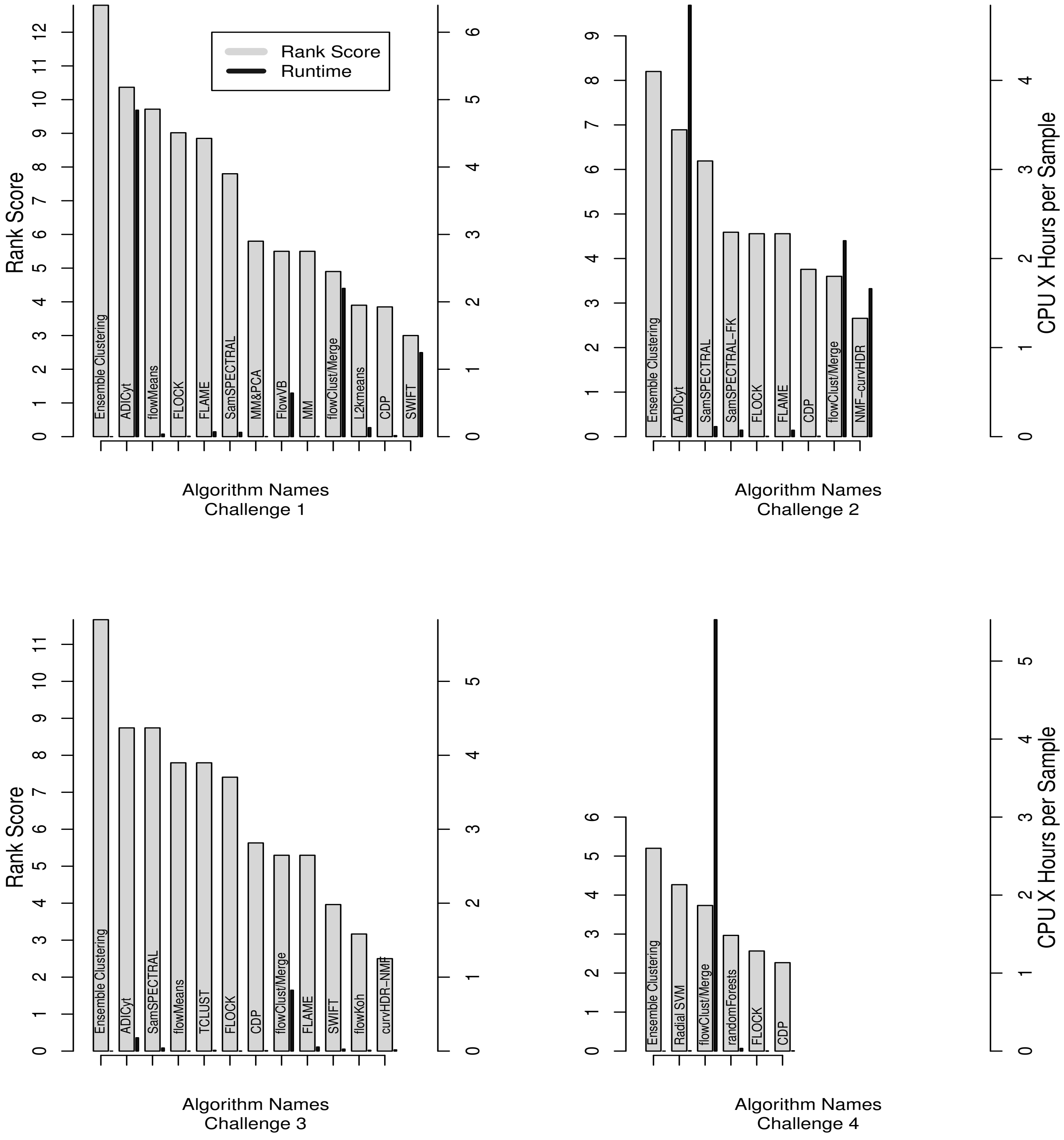
5 Supplementary Note 5: Future of FlowCAP Challenges

We identified several issues that remain to be addressed in future FlowCAP challenges. 1) For sample classification, participants were asked to provide discrete outputs (class names) rather than probabilities of each subject belonging to a particular class. This made it impossible to perform receiver operating characteristic (ROC) analysis and potentially decreased the robustness of the study. 2) For cell population identification, the data provided to the participants was pre-processed, which could potentially be a source of bias. For example, in some datasets the analysis was limited to the lymphocyte population, and other cells were manually excluded. Similarly, sometimes cell populations identified by the manual gates were indeed several cell populations based on other combinations of markers (an issue that in theory can be avoided through extensive back-gating). In some cases, for consistency with manual gating, algorithms were forced to process data that was improperly transformed. This was disadvantageous to the algorithms when, for example, artifactual clusters of cells were introduced [49]. For example, in some cases a log-transformation was used rather than the logicle resulting in a large number of events on the axes [49]. 3) Many of the algorithms evaluated in the sample classification challenges relied on matching cell populations across multiple samples. Several alternatives for this process have been proposed (*e.g.*, see [9,38,50,51] and also the *FlowCAP-II Algorithm Descriptions* section of the Supplementary Information), but the performance of population matching methods has never been compared objectively. 4) In retrospect, the data used for the sample classification challenges appeared to be either overly challenging (HEU vs. UE) or overly simple (AML and HVTN) for the algorithms. The analysis of these algorithms should be extended to evaluation using datasets with correlation structures that can be more challenging for these algorithms to reveal their potential shortcomings in more details. 5) Our preliminary results (post-hoc analysis of HVTN) suggest that computational methods can outperform humans in handling technical variation. This needs to be investigated in more detail by providing benchmarks of cross-institutional datasets with standardized panels (*e.g.*, those produced by the Human Immunology Project [52]) to design computational pipelines that are more robust to technical variation. 6) The runtimes of the cell population identification algorithms were measured using different hardware and software environments. While this provided an estimate of the time requirements, true direct comparison was not possible. In the sample classification challenges, the situation was further complicated by having separate training and testing procedures that often included visual exploration of the data by the algorithm developers. In future challenges, we intend to address this problem by introducing standardized interfaces and data-formats between the participating software and the evaluation pipeline so that the evaluation can be performed in a unified hardware/software setting. In addition to providing an objective comparison of time requirements, this will also facilitate both independent reproduction of the results and the adoption of these pipelines in biological and clinical laboratories.

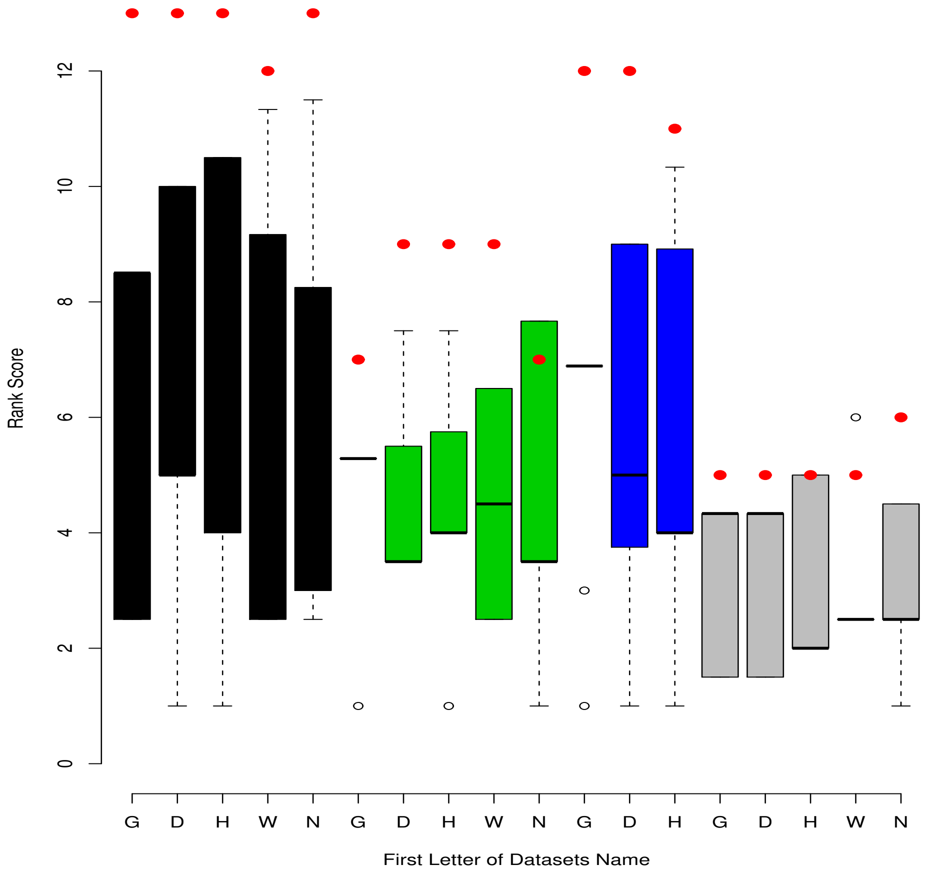
6 Supplementary Figures



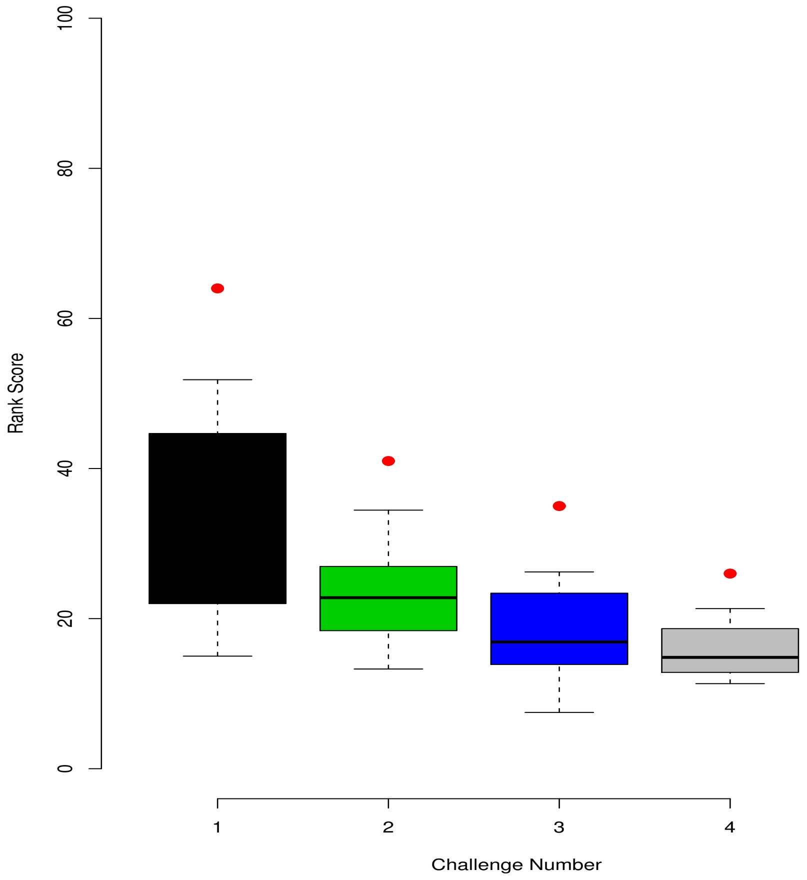
Supplementary Figure 1: Confidence intervals of the overall F-measures for each challenge. The top algorithm and the algorithms with overlapping CI's (right of the dashed line) are shown in green. The rest of the algorithms are shown in red.



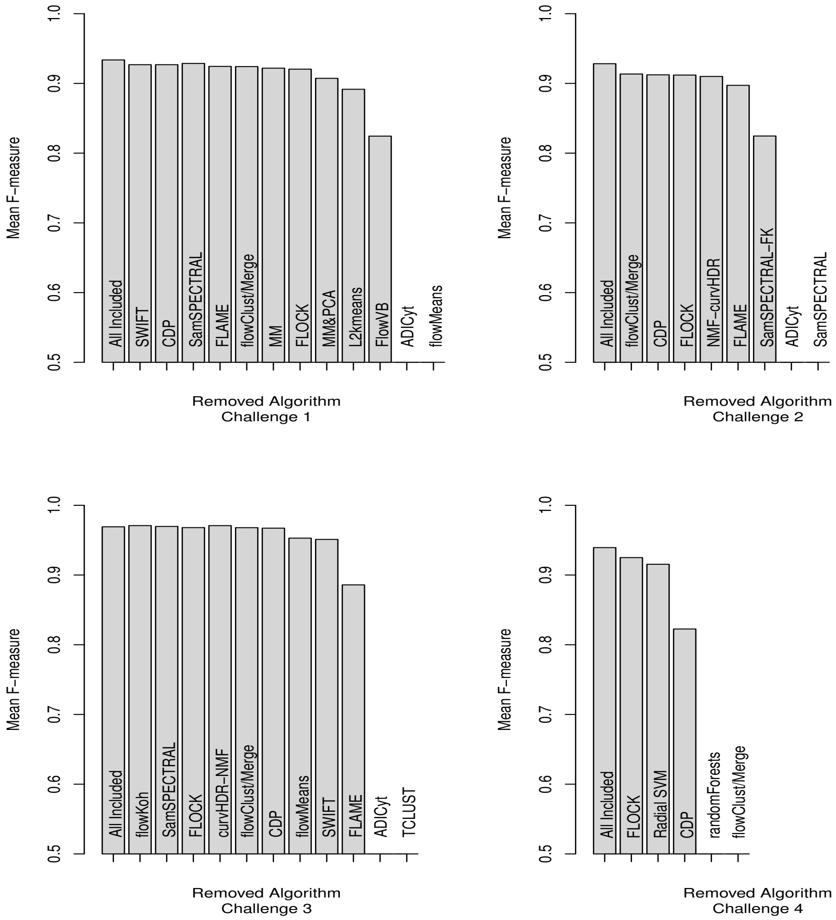
Supplementary Figure 2: Rank scores and runtimes (per CPU per sample) for each algorithm/challenge. The ensemble clustering's runtime is not included, but it would be close to the sum of the runtimes of all other algorithms.



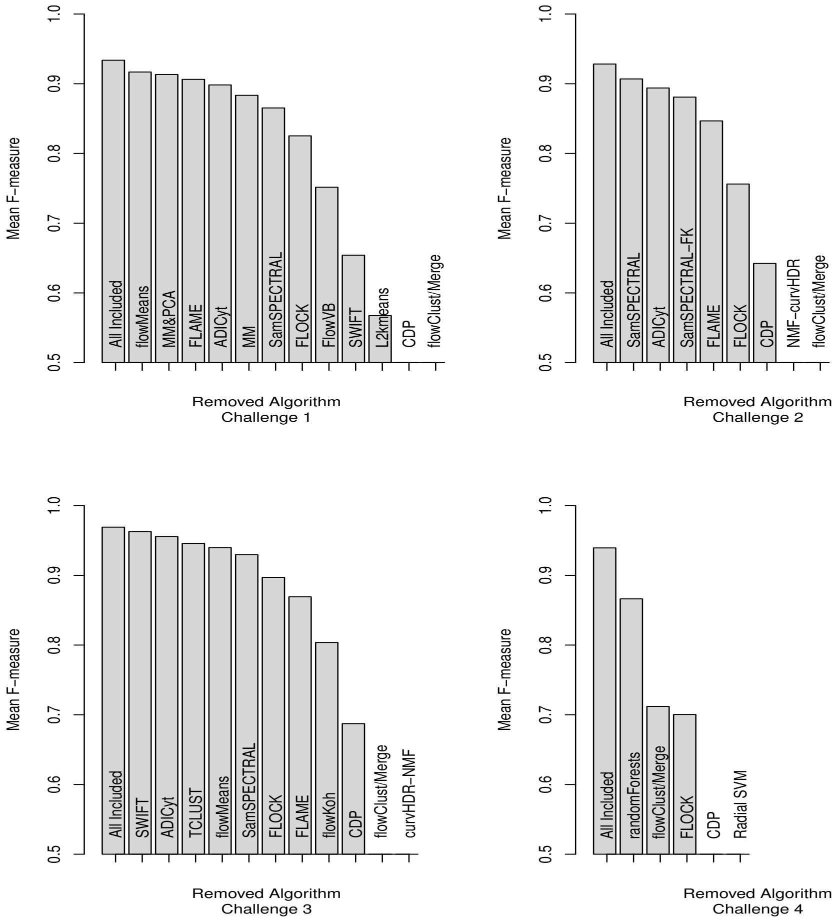
Supplementary Figure 3: Rank scores of all individual algorithms (box plots) are compared with the ensemble clustering (red dots) in each dataset/challenge.



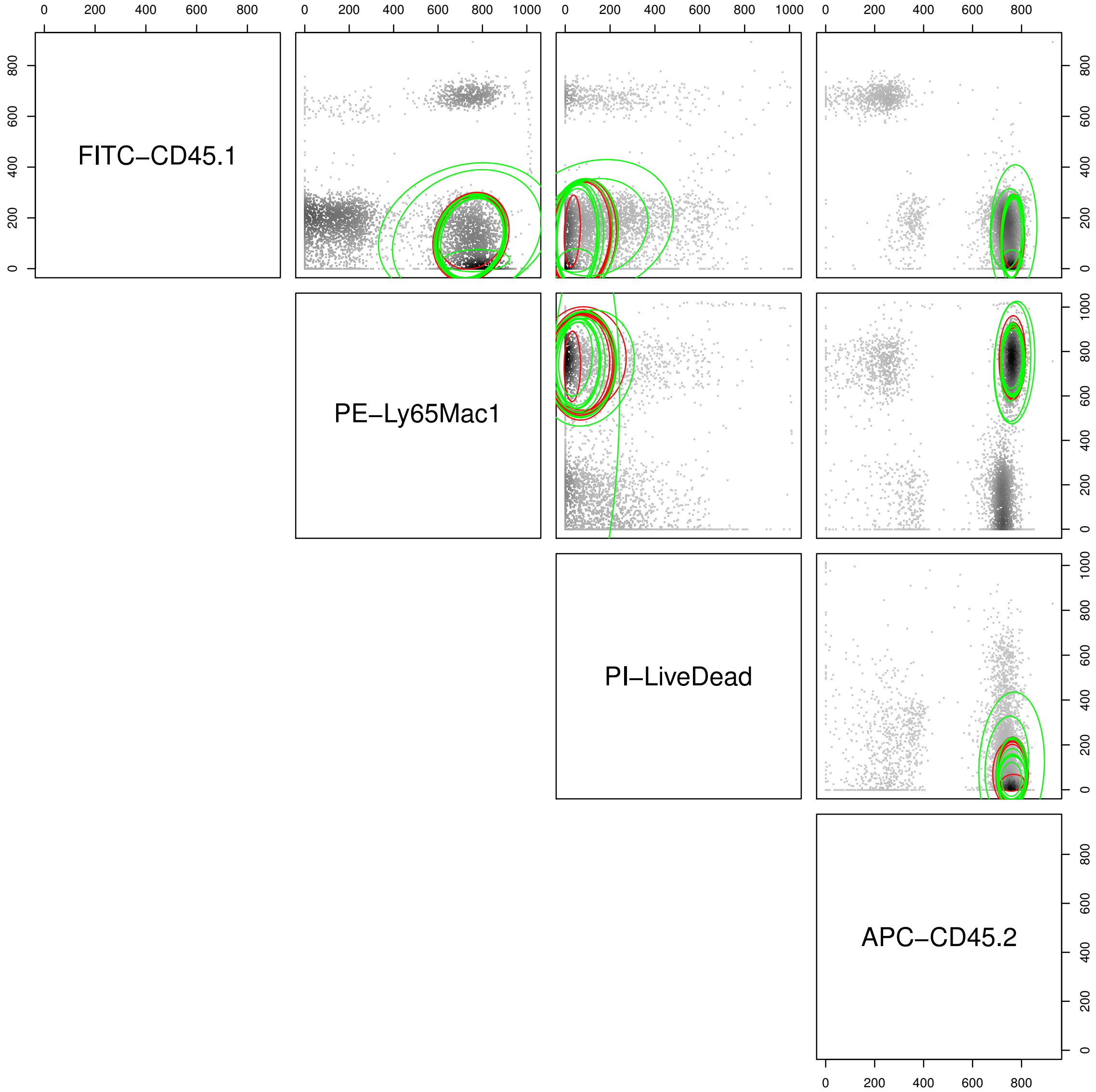
Supplementary Figure 4: Rank scores of all individual algorithms (box plots) are compared with the ensemble clustering (red dots) across all challenges.



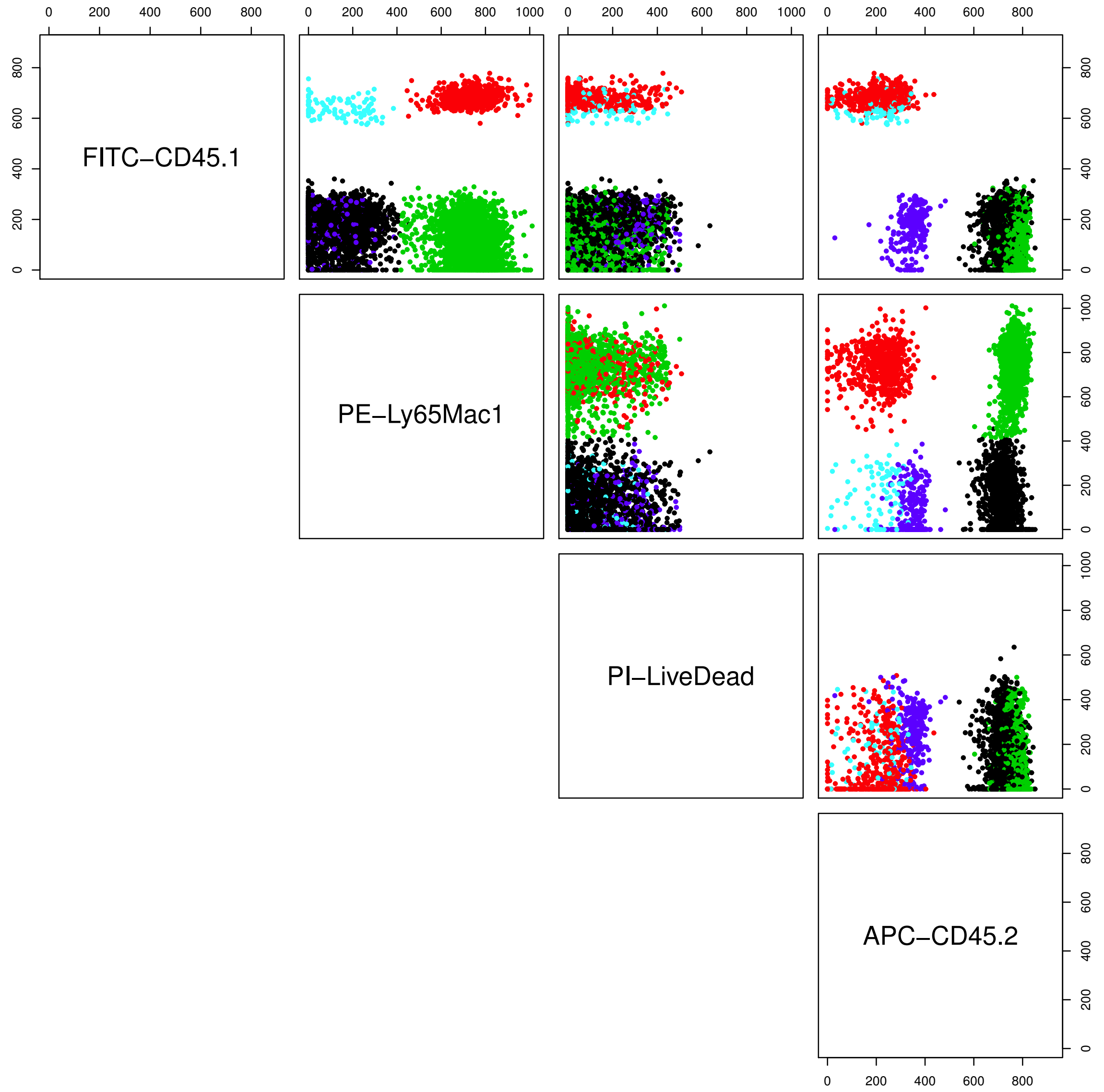
Supplementary Figure 5: Ablation analysis results. The change in F-measure as each algorithm was removed from the ensemble cluster in order of their relative contribution is shown, with the algorithms contributing less to the ensemble results removed first. The algorithm are listed in order of impact, from lowest to highest, on the F-measure value for each challenge, and the respective F-measure of the combined predictions indicated on the y-axis. Ensemble clustering for less than 3 algorithms is undefined for the CLUE package, therefore, the last two steps (where 2 and 1 algorithms are left, respectively) are not shown in this figure.



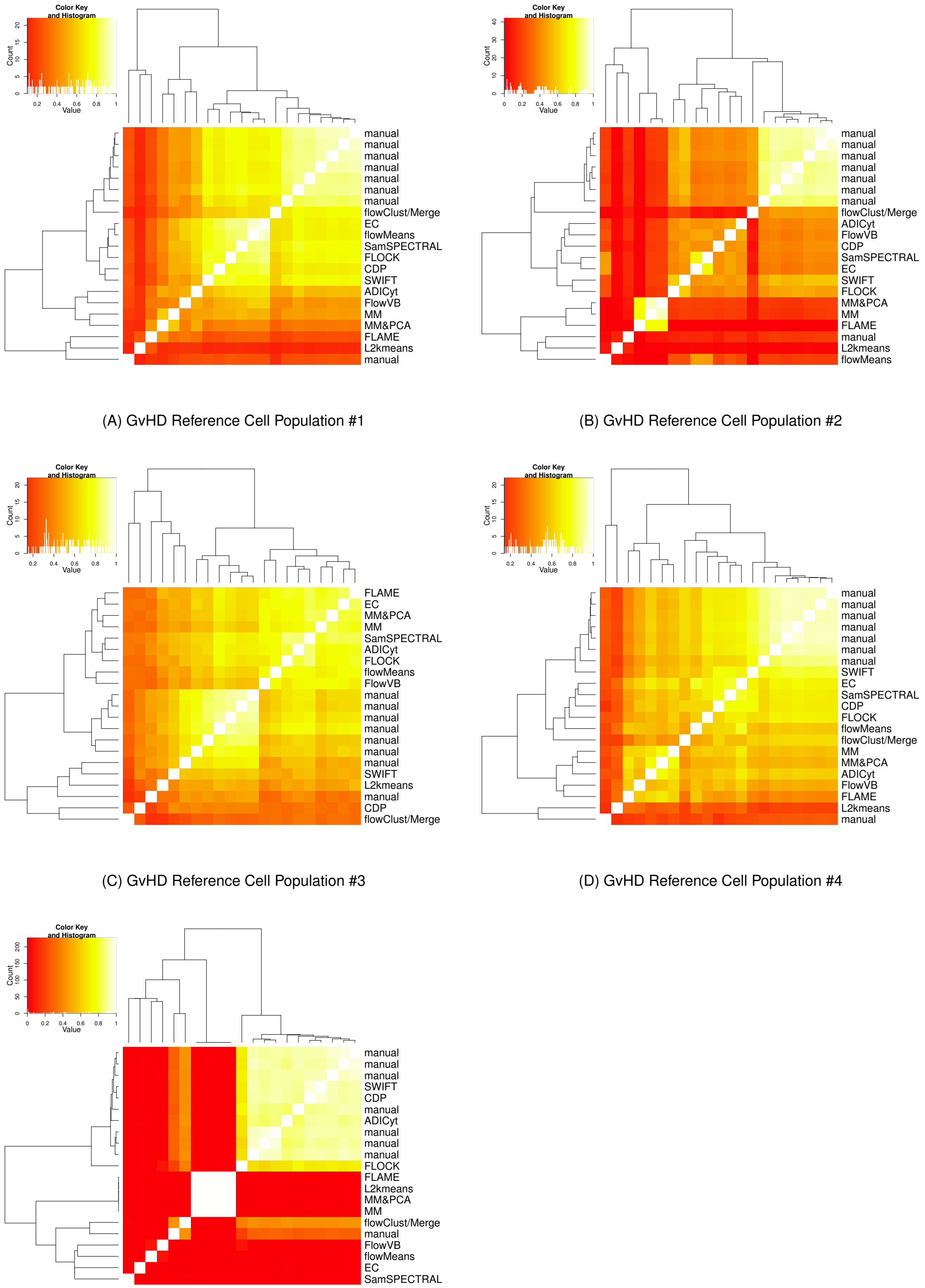
Supplementary Figure 6: Reversed ablation analysis results. The algorithm with maximum contribution at each step of the ablation analysis (for each challenge) and the respective F-measure of the combined predictions are listed from highest to lowest. Ensemble clustering for less than 3 algorithms is undefined for the CLUE package. Therefore, the last two steps (where 2 and 1 algorithms are left, respectively) are not shown in this figure.



Supplementary Figure 7: Scatter plot of Sample 26 of the HSCT dataset (the sample with maximum number of reference cell populations) for Cell Population #3 (green in Supplementary Figure 8) for which a relatively high agreement between all algorithms and manual gates have been observed (Figure 2). In this plot, algorithm results are partitioned with green ellipses and manual gating results are partitioned with red ellipses.



Supplementary Figure 8: Scatter plot of Sample 26 of the HSCT dataset (the sample with maximum number of reference cell populations), for visualization of all of the cell populations in the consensus among manual gates . Colors are as follow (can be matched to the panels of Figure 2): 1-black, 2-red, 3-green, 4-blue, and 5-cyan. The cyan population was consistently missed by six of the algorithms and one of the manual gates (Figure 2 Cell Population #5).



(A) GvHD Reference Cell Population #1

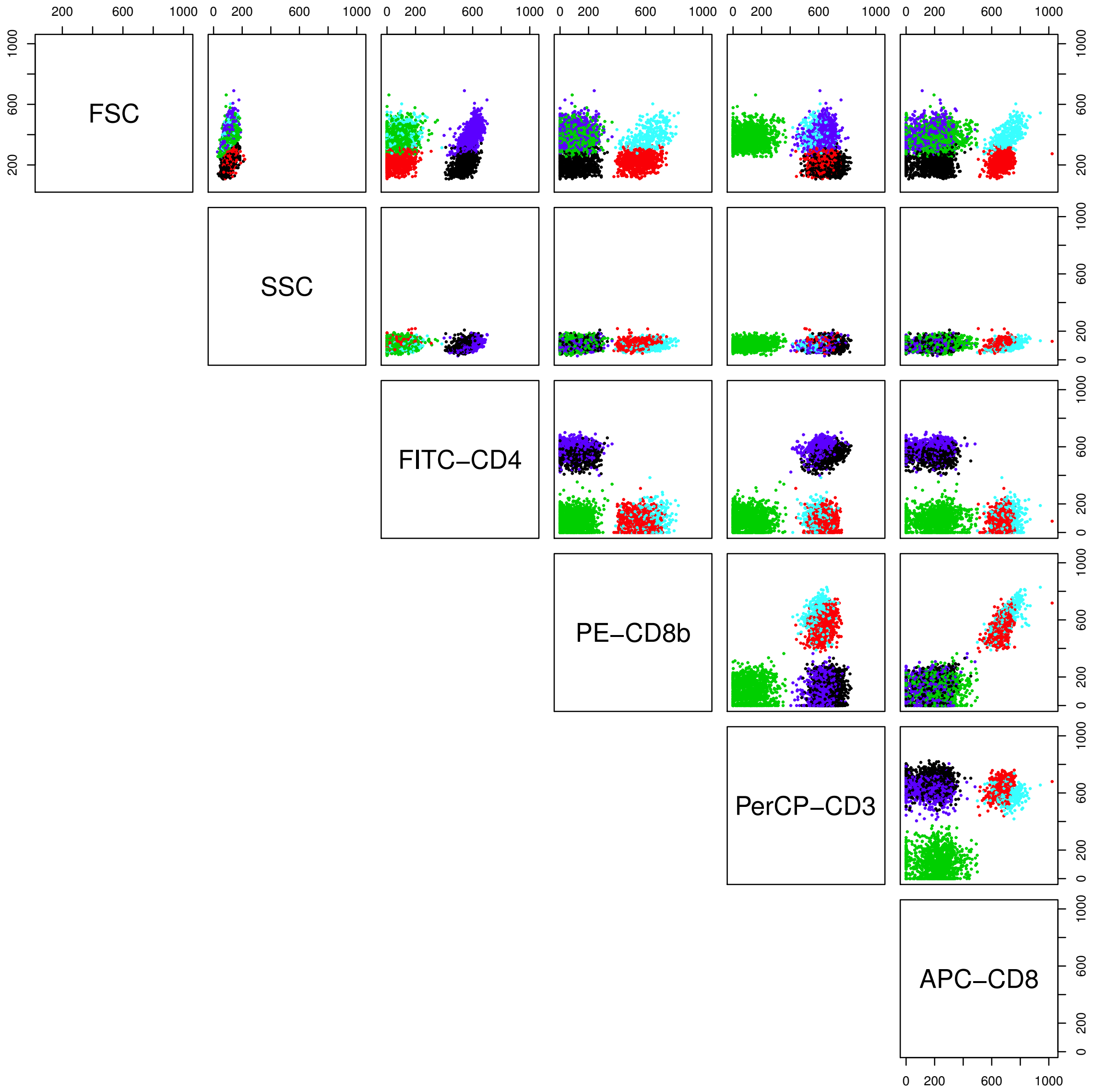
(B) GvHD Reference Cell Population #2

(C) GvHD Reference Cell Population #3

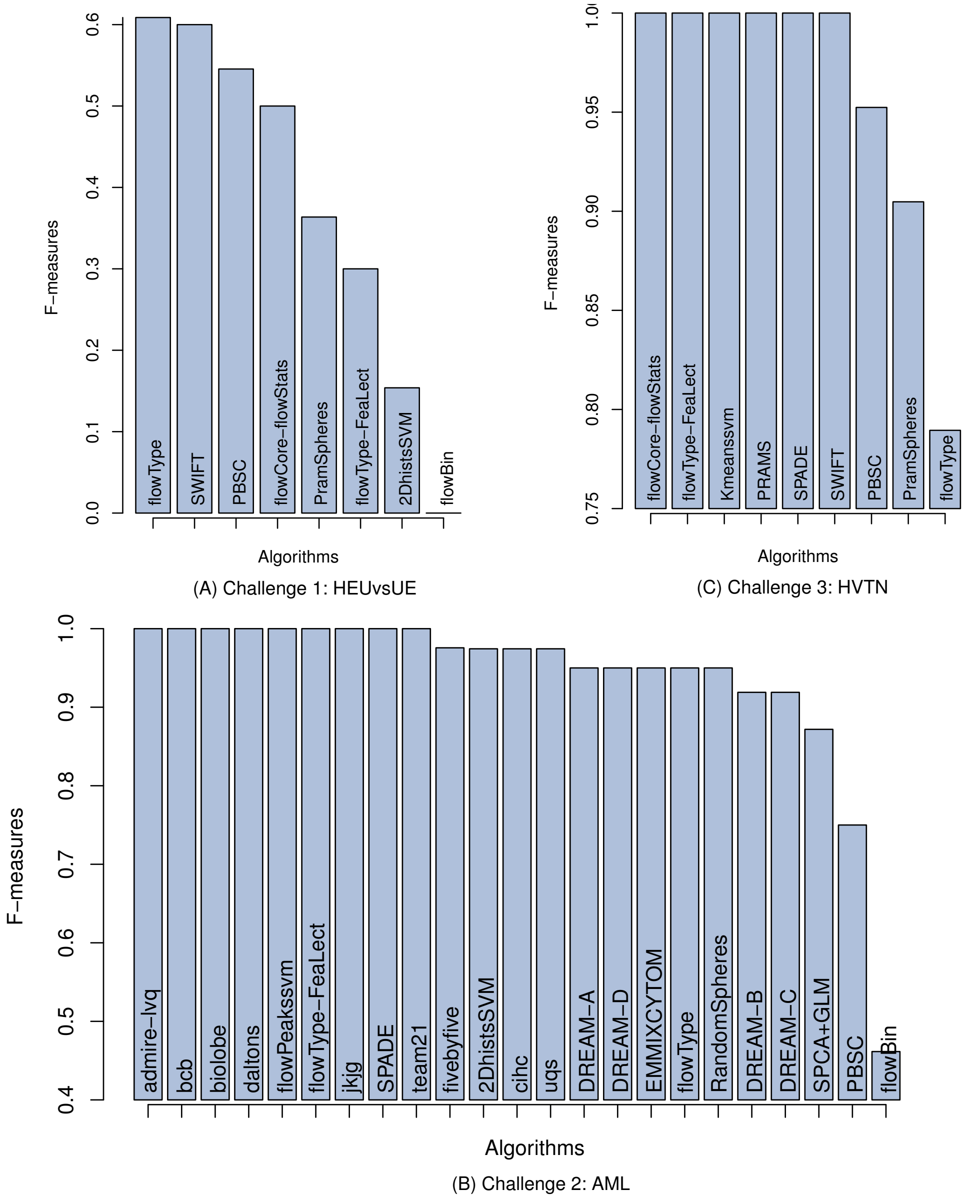
(D) GvHD Reference Cell Population #4

(E) GvHD Reference Cell Population #5

Supplementary Figure 9: Similar to Figure 2 for the GvHD dataset.



Supplementary Figure 10: Scatter plot of Sample 1 of the GvHD dataset (the sample with maximum number of reference cell populations). Colors are as follow (can be matched to the panels of Supplementary Figure 9): 1-black, 2-red, 3-green, 4-blue, and 5-cyan. The red population has been consistently missed by all of the algorithms and consistently identified by most of the manual gates (Supplementary Figure 9 Panel B). The only major difference between the red and the cyan population is in the forward scatter channel (FSC.H).



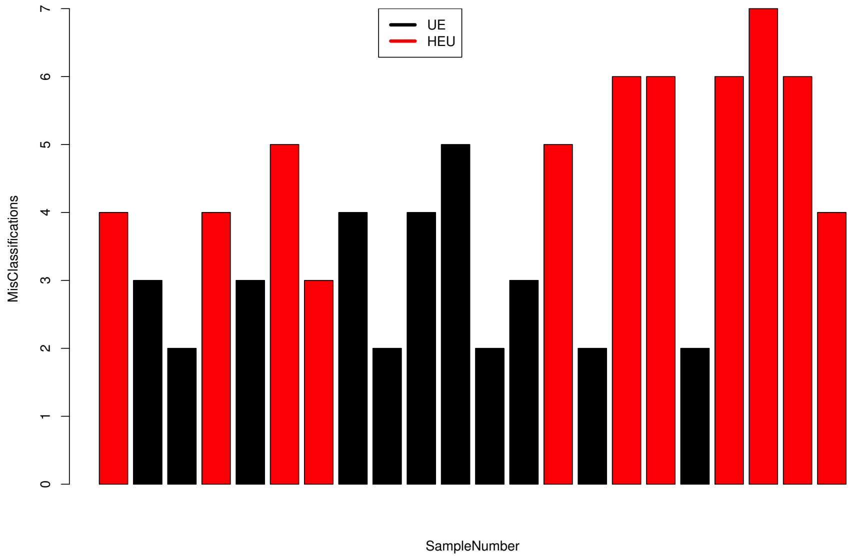
(A) Challenge 1: HEUvsUE

(C) Challenge 3: HVTN

(B) Challenge 2: AML

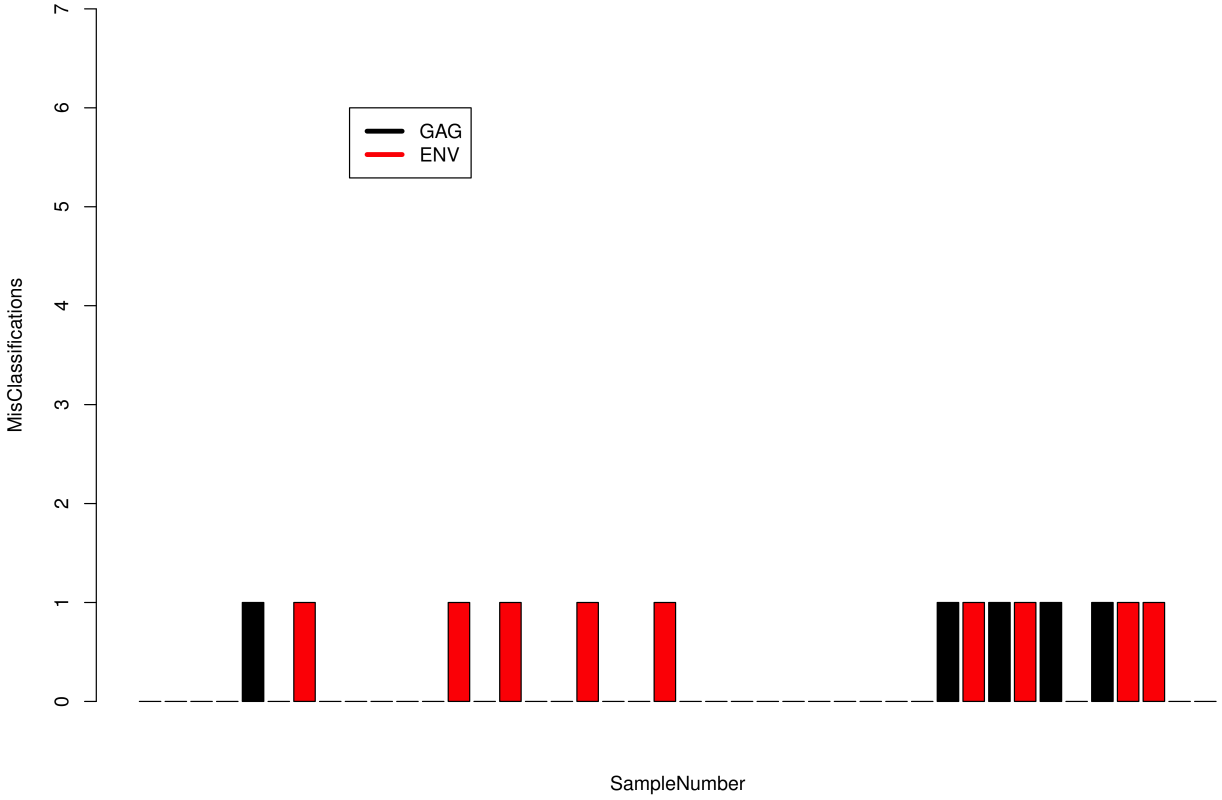
Supplementary Figure 11: Mean F-measures of the validation subset of the three datasets in FlowCAP-II.

Nature Methods: doi:10.1038/nmeth.2365

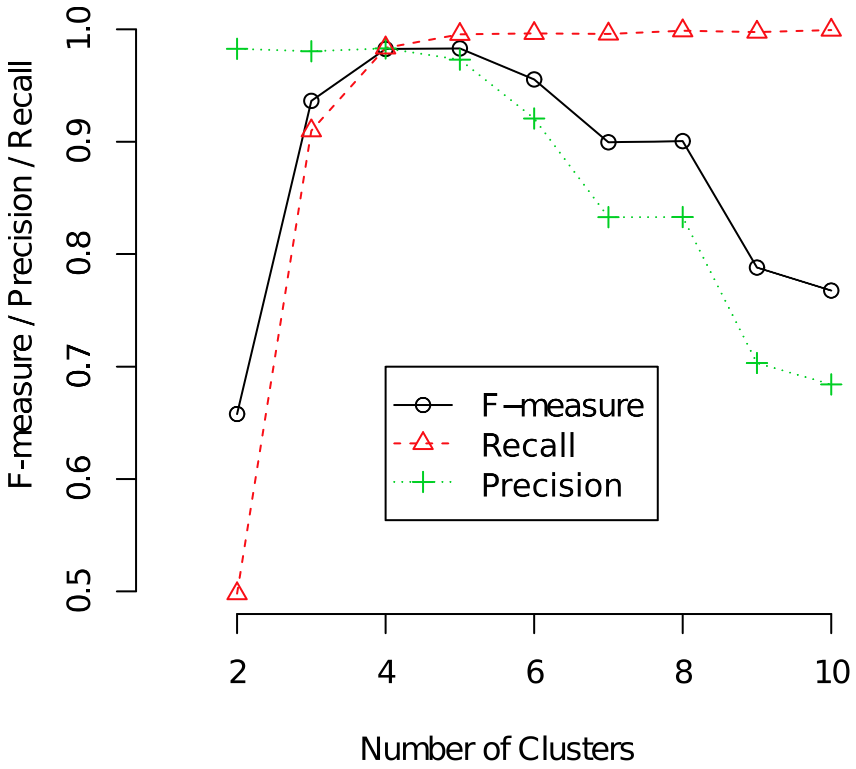


Supplementary Figure 12: Total number of mis-classifications for the samples in the test-set of the HEUvsUE dataset.

Nature Methods: doi:10.1038/nmeth.2365

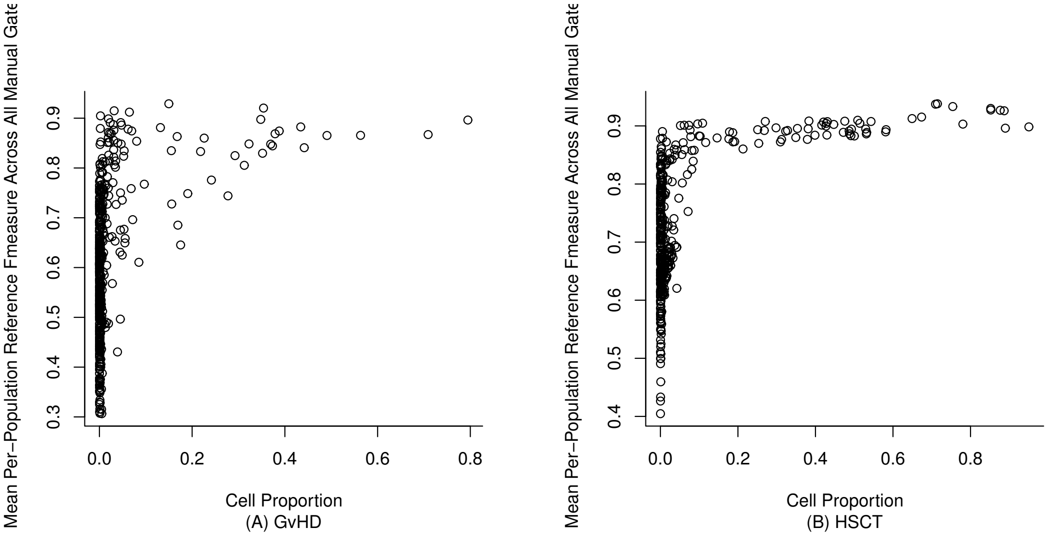


Supplementary Figure 13: Total number of mis-classifications for the samples in the test-set of the HVTN dataset.

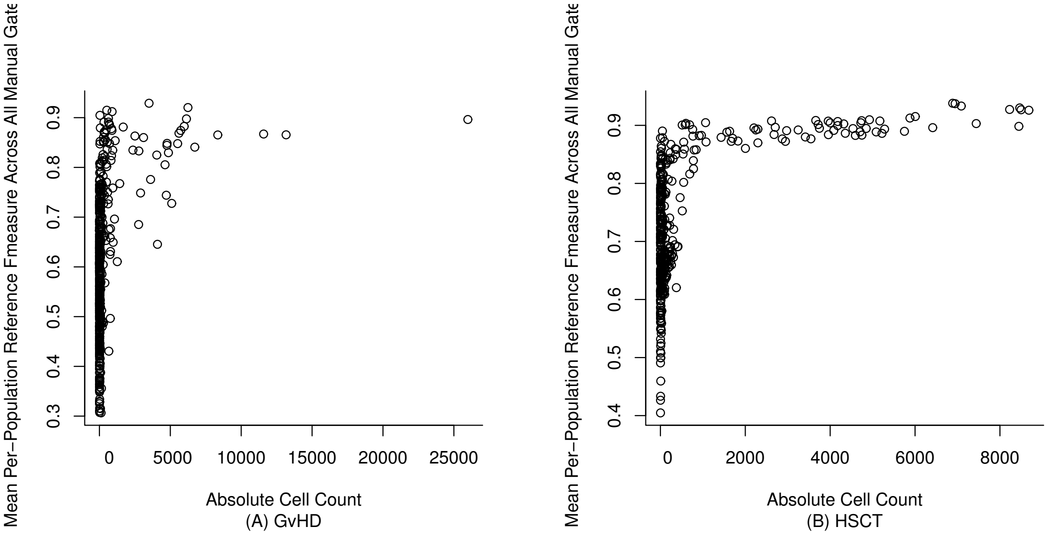


Supplementary Figure 14: Recall, precision, and F-measure of flowMeans for 2 to 10 clusters.

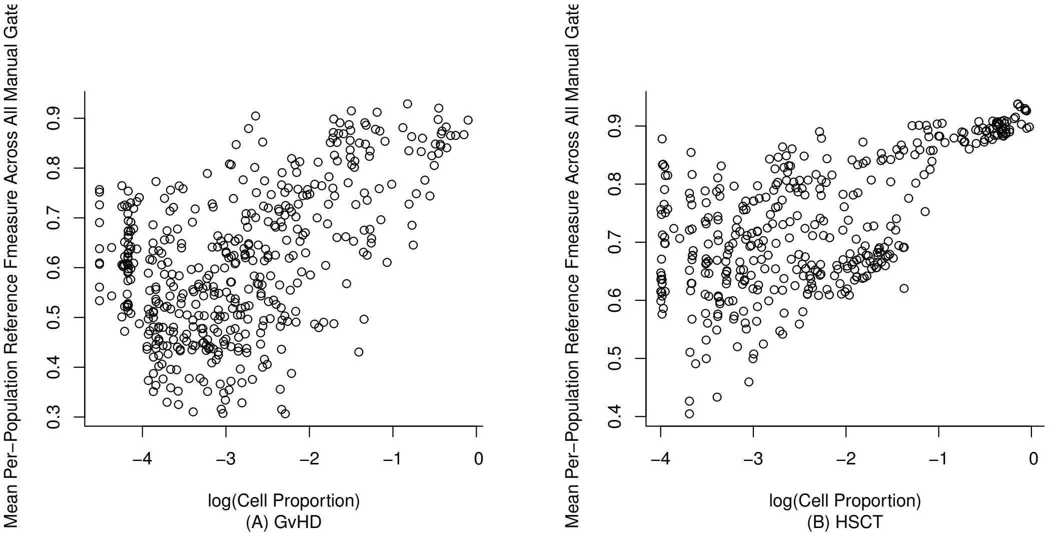
Nature Methods: doi:10.1038/nmeth.2365



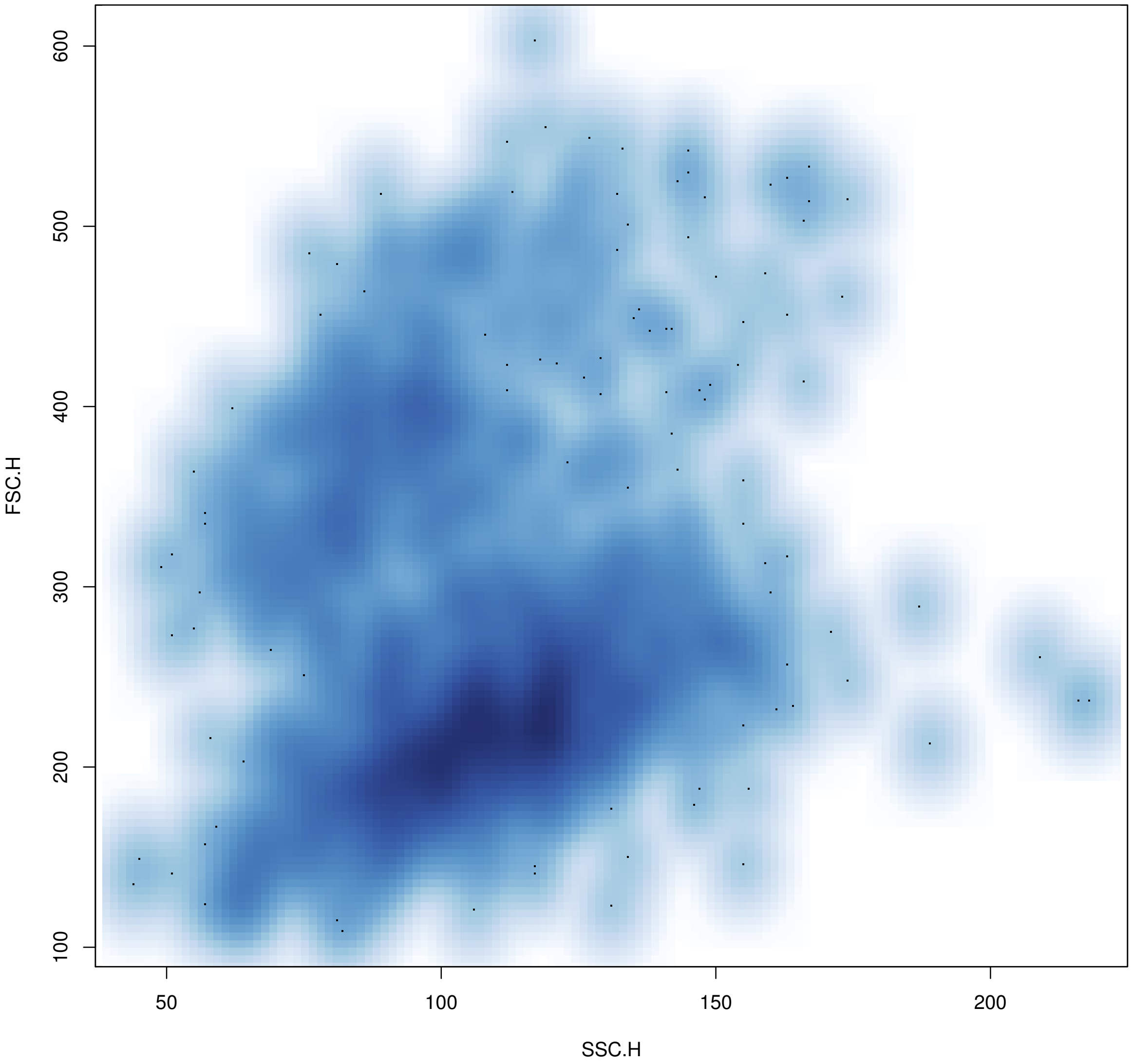
Supplementary Figure 15: Correlation between F-measure value and cell population size. These plots show the average F-measures versus the size of the cell population across the samples in the two datasets for all eight sets of manual gates. Generally, these data suggest that there is a stronger consensus among humans when the cell population is larger. Agreement among independent human gaters can also be found for some small cell populations but not for others.



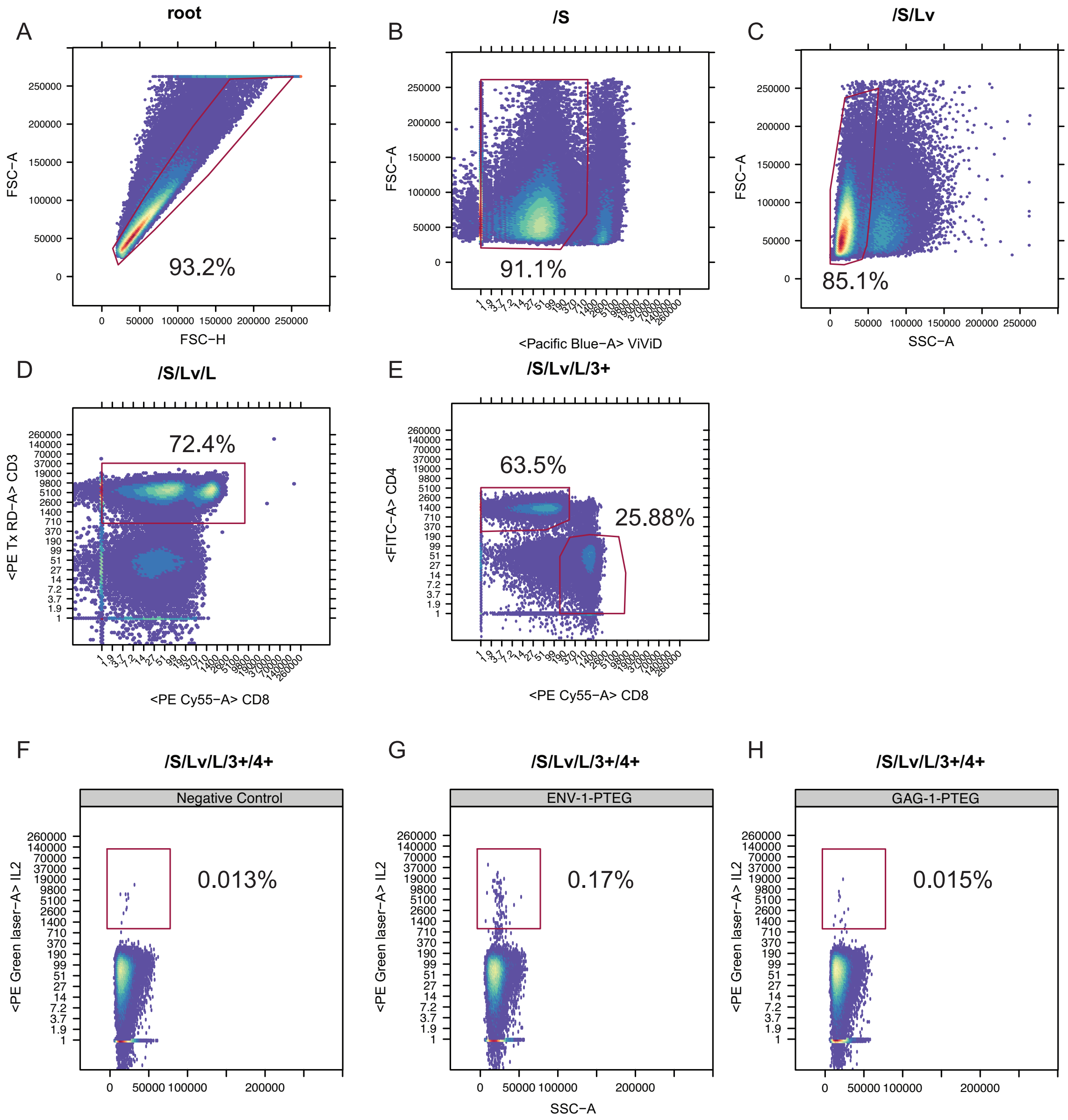
Supplementary Figure 16: Same as Supplementary Figure 15 using absolute cell count instead of cell proportion.



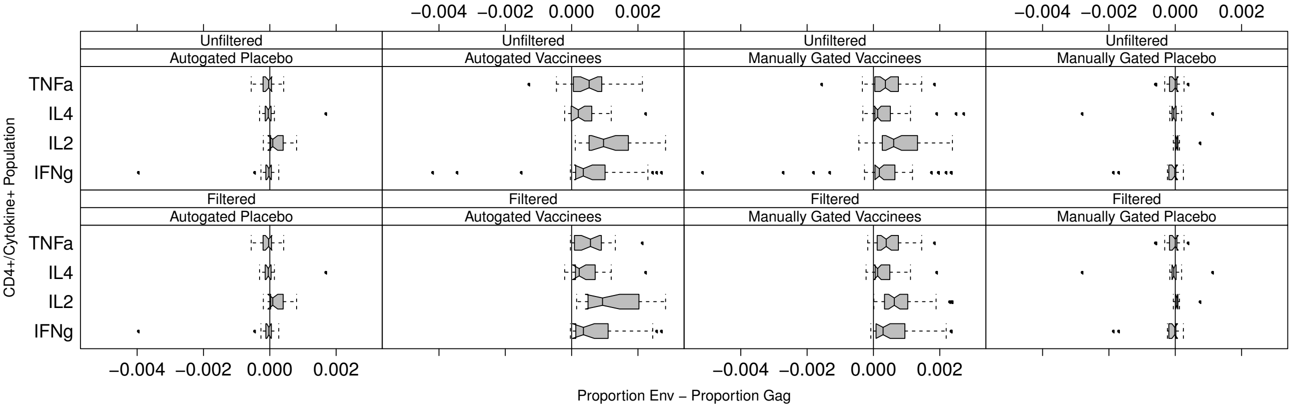
Supplementary Figure 17: Same as Supplementary Figure 15 on a $\log_{10}$ scale.



Supplementary Figure 18: Forward and side scatters of the red and cyan populations in Supplementary Figure 10 to confirm the existence of two different cell populations.



Supplementary Figure 19: An example of manual gating of FCS data from one individual from FlowCAP-II's HVTN challenge. An example of hierarchical manual gating of CD4 and CD8 T-cells is shown (A-E), with cell population proportions corresponding to the percent of cells in each gate. Gating of cytokine-positive, CD4^{+} T-cells is shown for the F) negative control, G) ENV-1-PTEG stimulation, H) GAG-1-PTEG stimulated samples from the same individual, with corresponding cell sub-population proportions. Typical manual analysis requires such gating of each stimulation and control sample from each individual and comparison of the cytokine-positive T-cell proportions (for CD4 and CD8) for all measured cytokines.



Supplementary Figure 20: Comparison of automated and manual gating on the paired difference between Env and Gag stimulated samples expressing different CD4 T–cell subpopulations. Unfiltered data, including samples from mismatched plates / runs (top row) gated using automated methods on the placebo group (first column), automated methods on the vaccinee group (second column), manual methods on the vaccinee group (third column), or manual methods on the placebo group (fourth column), as well as with samples from mismatched plates filtered out (bottom row).

7 Supplementary Tables

Supplementary Table 1: Ranking of the algorithms based on the original and refined manual gates. The algorithms that are not significantly different from the top algorithm are bolded (see Table 2 in the manuscript for the confidence intervals).

GvHD		HSCT	
Original	Refined	Original	Refined
Ensemble Clustering	**Ensemble Clustering**	**Ensemble Clustering**	**Ensemble Clustering**
flowMeans	**ADICyt**	**FLAME**	**FLAME**
SamSPECTRAL	**flowMeans**	**ADICyt**	**ADICyt**
FLAME	**FLAME**	**flowMeans**	**MM&PCA**
FlowVB	**SamSPECTRAL**	**MM&PCA**	**flowMeans**
FLOCK	**FlowVB**	FLOCK	FLOCK
MM&PCA	**FLOCK**	SamSPECTRAL	FlowVB
MM	**MM**	**flowClust/Merge**	SamSPECTRAL
ADICyt	**MM&PCA**	FlowVB	flowClust/Merge
flowClust/Merge	SWIFT	MM	L2kmeans
L2kmeans	flowClust/Merge	L2kmeans	MM
SWIFT	L2kmeans	SWIFT	SWIFT
CDP	CDP	CDP	CDP

Supplementary Table 2: Results of the independent analysis by DREAM. Four of the algorithm names are not disclosed and therefore have been replaced with DREAM-A to DREAM-D.

Team	Pearson	Prec	Rec	MCC	JSC	Score	Rank	Rank Among Best Performers
team21	1.00	1.00	1.00	1.00	1.00	1.00	1	1
BCB	0.99	1.00	1.00	1.00	1.00	1.00	1	2
Admire-LVQ	0.97	1.00	1.00	1.00	1.00	1.00	1	3
JKJG	0.97	1.00	1.00	1.00	1.00	1.00	1	4
Daltons	0.97	1.00	1.00	1.00	1.00	1.00	1	5
biolobe	0.88	1.00	1.00	1.00	1.00	1.00	1	6
UQS	0.97	0.95	0.95	0.94	0.90	0.94	7	-
FiveByFive	0.95	0.95	0.95	0.94	0.90	0.94	7	-
DREAM-D	0.94	0.95	0.95	0.94	0.90	0.94	7	-
DREAM-A	0.89	0.95	0.95	0.94	0.90	0.94	7	-
cihc	0.65	0.95	0.95	0.94	0.90	0.94	7	-
DREAM-C	0.93	0.90	0.90	0.89	0.82	0.88	12	-
DREAM-B	0.89	0.90	0.90	0.89	0.82	0.88	12	-
SPCA+GLM	0.87	0.85	0.85	0.83	0.74	0.82	14	-

Supplementary Table 3: Contact information of the participating teams.

Algorithm Name	Contact Person	Email
ADICyt	Peter Majek	majek@adinis.sk
CDP	Cliburn Chan	cliburn.chan@duke.edu
FLAME	Kui Wang	kwang@maths.uq.edu.au
FLOCK	Yu Qian	qianyu_cs@yahoo.com
flowClust/Merge	Greg Finak	Greg.Finak@ircm.qc.ca
flowKoh	Radina Nikolic	rnikolic@bccrc.ca
flowMeans	Nima Aghaeepour	naghaeep@bccrc.ca
FlowVB	Hannes Bretschneider	habretschneider@gmail.com
L2kmeans	Faysal El Khettabi	fkhettabi@bccrc.ca
MM, MM&PCA	Istvan Sugar	istvansugar0@gmail.com
NMF-curvHDR	Joe Maisog	bravas02@gmail.com
Radial SVM	John Quinn	john@treestar.com
SamSPECTRAL	Habil Zare	zare@u.washington.edu
SWIFT	Iftekhar Naim	naim@ece.rochester.edu
2DhistSVM	Maria Chikina	mchikina@gmail.com
admire-lvq	Michael Biehl	m.biehl@rug.nl
BCB, SPADE	Peng Qiu	pqiu@mdanderson.org
biolobe	Marc Strickert	strickert@informatik.uni-siegen.de
cihc	Miron Kursa,	m.kursa@icm.edu.pl
daltons	Tapio Manninen	tapio.manninen@tut.fi
EMMIXCYTOM and uqs	Geoff McLachlan	g.mclachlan@uq.edu.au
DREAM–A	Anonymized	-
DREAM–B	Anonymized	-
DREAM–C	Anonymized	-
DREAM–D	Anonymized	-
FiveByFive	Mark Maienschein-Cline	mmaiensc@gmail.com
flowBin	Kieran O'Niell	koneill@bccrc.ca
FlowCore-flowStats	Greg Finak	gfinak@fhcrc.org
flowPeakssvm and Kmeanssvm	Yongchao Ge	yongchao.ge@mssm.edu
flowType, flowType-FeaLect	Nima Aghaeepour	naghaeep@bccrc.ca
JKJG	Jens Keilwagen	Jens.Keilwagen@ipk-gatersleben.de
PBSC	Yu Qian	Yu.Qian@utsouthwestern.edu
PRAMS	Robert Bruggner	bruggner@stanford.edu
Pram Spheres, Random Spheres	Miron B. Kursa	M.Kursa@icm.edu.pl
SPADE	Peng Qiu	PQiu@mdanderson.org
SPCA+GLM	Madhu Bhattacharjee	chhanda.bhatta@gmail.com
SWIFT	Gaurav Sharma	gsharma@ece.rochester.edu
team21	Jose Vilar	j.vilar@ikerbasque.org
uqs	Geoff McLachlan	g.mclachlan@uq.edu.au

Supplementary Table 4: Summary of the description of the cell population identification datasets.

Dataset	#Samples	#Events	Analyte	Detector	Reporter	Provided By
DLBCL	30	5,000	CD3	Anti-CD3	CY5	BCCRC
			CD5	Anti-CD5	FITC	
			CD19	Anti-CD19	PE	
WNV	13	100,000	IFNγ	Anti-IFNγ	PEA	McMaster
			CD3	Anti-CD3	PECy5	
			CD4	Anti-CD4	PECy7	
			IL17	Anti-IL17	APC	
			CD8	Anti-CD8	AlexaFluor700	
			Free Amines	NA	CFSE	
ND	30	17,000		Proprietary	FITC	Amgen
				Proprietary	PerCPCy5	
				Proprietary	PacificBlue	
				Proprietary	PacificOrange	
			CD56	Anti-CD56	Qdot605	
				Proprietary	APC	
			CD8	Anti-CD8	Alexa700	
				Proprietary	PE	
			CD45	Anti-CD45	PECy5	
			CD3/CD14	Anti-CD3/CD14	PECy7	
HSCT	30	10,000	CD45.1	Anti-CD45.1	FITC	BCCRC
			Ly65/Mac1	Anti-Ly65/Mac1	PE	
			Dead Cells	NA	PI	
			CD45.2	Anti-CD45.2	APC	
GvHD	12	14,000	CD4	Anti-CD4	FITC	BCCRC
			CD8b	Anti-CD8b	PE	&
			CD3	Anti-CD3	PerCP	TreeStar
			CD8	Anti-CD8	APC	

Supplementary Table 5: List of reporter and analytes for the HVTN, HEUvsUE, and all nine panels of the AML datasets.

HEUvsUE		HVTN		AML 1		AML 2		AML 3	
Channel	Reagent	Channel	Reagent	Channel	Reagent	Channel	Reagent	Channel	Reagent
FSC-A		FSC-A		FSC	FSC	FSC	FSC	FSC	FSC
SSC-A		FSC-H		SSC	SSC	SSC	SSC	SSC	SSC
FITC-A	IFNa	SSC-A		FL1	IgG1-FITC	FL1	Kappa-FITC	FL1	CD7-FITC
PE-A	CD123	FITC-A	CD4	FL2	IgG1-PE	FL2	Lambda-PE	FL2	CD4-PE
PerCP-Cy5-5-A	MHCII	Pacific Blue-A	ViViD	FL3	CD45-ECD	FL3	CD45-ECD	FL3	CD45-ECD
PE-Cy7-A	CD14	Alexa 680-A	TNFa	FL4	IgG1-PC5	FL4	CD19-PC5	FL4	CD8-PC5
APC-A	CD11c	APC-A	IL4	FL5	IgG1-PC7	FL5	CD20-PC7	FL5	CD2-PC7
APC-Cy7-A	IL6	PE Cy7-A	IFNg						
Pacific Blue-A	IL12	PE Cy55-A	CD8						
Alex 700-A	TNFa	PE Tx RD-A	CD3						
		PE Green laser-A	IL2						
AML 4		AML 5		AML 6		AML 7		AML 8	
Channel	Reagent	Channel	Reagent	Channel	Reagent	Channel	Reagent	Channel	Reagent
FSC	FSC	FSC	FSC	FSC	FSC	FSC	FSC	FSC	FSC
SSC	SSC	SSC	SSC	SSC	SSC	SSC	SSC	SSC	SSC
FL1	CD15-FITC	FL1	CD14-FITC	FL1	HLA DR-FITC	FL1	CD5-FITC	FL1	FL1
FL2	CD13-PE	FL2	CD11c-PE	FL2	CD117-PE	FL2	CD19-PE	FL2	FL2
FL3	CD45-ECD	FL3	CD45-ECD	FL3	CD45-ECD	FL3	CD45-ECD	FL3	7AAD
FL4	CD16-PC5	FL4	CD64-PC5	FL4	CD34-PC5	FL4	CD3-PC5	FL4	FL4
FL5	CD56-PC7	FL5	CD33-PC7	FL5	CD38-PC7	FL5	CD10-PC7	FL5	FL5

8 References

[1] G. Finak, A. Bashashati, R. Brinkman, and R. Gottardo. Merging mixture components for cell population identification in flow cytometry. *Advances in Bioinformatics*, 2009, 2009.

[2] K. Lo, R.R. Brinkman, and R. Gottardo. Automated gating of flow cytometry data via robust model-based clustering. *Cytometry Part A*, 73(4):321–332, 2008.

[3] C. Chan, F. Feng, J. Ottinger, D. Foster, M. West, and T.B. Kepler. Statistical mixture modeling for cell subtype identification in flow cytometry. *Cytometry Part A*, 73(8):693–701, 2008.

[4] I. Manolopoulou, C. Chan, and M. West. Selection sampling from large data sets for targeted inference in mixture modeling. *Bayesian analysis (Online)*, 5(3):1, 2010.

[5] C. Chan, L. Lin, J. Frelinger, V. Hérbert, D. Gagnon, C. Landry, R.P. Sékaly, J. Enzor, J. Staats, K.J. Weinhold, et al. Optimization of a highly standardized carboxyfluorescein succinimidyl ester flow cytometry panel and gating strategy design using discriminative information measure evaluation. *Cytometry Part A*, 2010.

[6] M.A. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West. Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, 19(2):419–438, 2010.

[7] F. Jacob, K. Thomas, and C. Chan. Flow: Statistics, visualization and informatics for flow cytometry. *Source Code for Biology and Medicine*, 3, 2008.

[8] J. Frelinger, J. Ottinger, C. Gouttefangeas, and C. Chan. Modeling flow cytometry data for cancer vaccine immune monitoring. *Cancer Immunology, Immunotherapy*, 59(9):1435–1441, 2010.

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

[10] L. Kaufman, P.J. Rousseeuw, et al. *Finding groups in data: an introduction to cluster analysis*, volume 39. Wiley Online Library, 1990.

[11] M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, and J.P. Mesirov. Genepattern 2.0. *Nature genetics*, 38(5):500–501, 2006.

[12] Y. Qian, C. Wei, F. Eun-Hyung Lee, J. Campbell, J. Halliley, J.A. Lee, J. Cai, Y.M. Kong, E. Sadat, E. Thomson, et al. Elucidation of seventeen human peripheral blood B-cell subsets and quantification of the tetanus response using a density-based method for the automated identification of cell populations in multidimensional flow cytometry data. *Cytometry Part B: Clinical Cytometry*, 78(S1):S69–S82, 2010.

[13] A.D. Diehl, A.D. Augustine, J.A. Blake, L.G. Cowell, E.S. Gold, T.A. Gondré-Lewis, A.M. Masci, T.F. Meehan, P.A. Morel, A. Nijnik, et al. Hematopoietic cell types: Prototype for a revised cell ontology. *Journal of biomedical informatics*, 44(1):75–79, 2011.

[14] Y. Qian, Y. Liu, J. Campbell, E. Thomson, Y.M. Kong, and R.H. Scheuermann. FCSTrans: An open source software system for FCS file conversion and data transformation. *Cytometry Part A*, 2012.

[15] T. Kohonen. Self-organizing neural projections. *Neural networks*, 19(6-7):723–733, 2006.

[16] R. Wehrens and L.M.C. Buydens. Self-and super-organizing maps in r: the kohonen package. *Journal of Statistical Software*, 21(5):19, 2007.

[17] N. Aghaeepour, R. Nikolic, H. H. Hoos, and R.R. Brinkman. Rapid Cell Population Identification in Flow Cytometry Data. *Cytometry Part A*, 79(1):6–13, 2011.

[18] Cédric Archambeau and Michel Verleysen. Robust bayesian clustering. *Neural Networks*, 20(1):129 – 138, 2007.

[19] I. Sugar and S. Sealfon. Misty Mountain clustering: application to fast unsupervised flow cytometry gating. *BMC Bioinformatics*, 11(1):502, 2010.

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

[21] V. Kecman. *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. The MIT press, 2001.

[22] J. Toedling, P. Rhein, R. Ratei, L. Karawajew, and R. Spang. Automated in-silico detection of cell populations in flow cytometry readouts and its application to leukemia disease monitoring. *BMC Bioinformatics*, 7(1):282, 2006.

[23] J. Quinn, P.W. Fisher, R.J. Capocasale, R. Achuthanandam, M. Kam, P.J. Bugelski, and L. Hrebien. A statistical pattern recognition approach for determining cellular viability and lineage phenotype in cultured cells and murine bone marrow. *Cytometry Part A*, 71(8):612–624, 2007.

[24] I Tsochantaridis, T Hofmann, T Joachims, and Y Altun. Support vector machine learning for interdependent and structured output spaces. In *Intl Conf. on Machine Learning*, 2004.

[25] P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[26] M. Strickert, A.J. Soto, and G.E. Vazquez. Adaptive matrix distances aiming at optimum regression subspaces. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning-ESANN*, pages 93–98, 2010.

[27] R.E. Moore. Ranking income distributions using the geometric mean and a related general measure. *Southern Economic Journal*, pages 69–75, 1996.

[28] R. Breitling, P. Armengaud, A. Amtmann, and P. Herzyk. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS letters*, 573(1):83–92, 2004.

[29] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[30] C. Furlanello, M. Serafini, S. Merler, and G. Jurman. Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC Bioinformatics*, 4(1):54, 2003.

[31] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[32] A.L. Muratov and O.Y. Gnedin. Modeling the metallicity distribution of globular clusters. *The Astrophysical Journal*, 718:1266, 2010.

[33] C. Mason, R. Hanson, V. Ossowski, L. Bian, L. Baier, J. Krakoff, and C. Bogardus. Bimodal distribution of rna expression levels in human skeletal muscle tissue. *BMC Genomics*, 12(1):98, 2011.

[34] P. Baldi, S. Brunak, Y. Chauvin, C.A.F. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412, 2000.

[35] M. Maienschein-Cline, A.R. Dinner, W.S. Hlavacek, and F. Mu. Improved predictions of transcription factor binding sites using physicochemical features of dna. *Nucleic Acids Research*, 2012.

[36] A.L. Bauer, W.S. Hlavacek, P.J. Unkefer, and F. Mu. Using sequence-specific chemical and structural properties of dna to predict transcription factor binding sites. *PLoS computational biology*, 6(11):e1001007, 2010.

[37] C.E. Pedreira, E.S. Costa, S. Barrena, Q. Lecrevisse, J. Almeida, J.J.M. van Dongen, and A. Orfao. Generation of flow cytometry data files with a potentially infinite number of dimensions. *Cytometry Part A*, 73(9):834–846, 2008.

[38] G. Lee, W. Finn, and C. Scott. Statistical file matching of flow cytometry data. *Journal of Biomedical Informatics*, 2011.

[39] A.K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[40] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[41] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

[42] Y. Ge and S.C. Sealfon. flowPeaks: a fast unsupervised clustering for flow cytometry data via K-means and density peak finding. *Bioinformatics*, 8(15):2052–2058, 2012.

[43] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12, pages 185–208. MIT Press, 1999.

[44] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[45] N. Aghaeepour, P.K. Chattopadhyay, A. Ganesan, K. O'Neill, H. Zare, A. Jalali, H.H. Hoos, M. Roederer, and R.R. Brinkman. Early immunologic correlates of hiv protection can be identified from computational analysis of complex multivariate t-cell flow cytometry assays. *Bioinformatics*, 2012.

[46] H. Zare, A. Bashashati, R. Kridel, N. Aghaeepour, G. Haffari, J.M. Connors, R.D. Gascoyne, A. Gupta, R.R. Brinkman, and A.P. Weng. Automated analysis of multidimensional flow cytometry data improves diagnostic accuracy between mantle cell lymphoma and small lymphocytic lymphoma. *American Journal of Clinical Pathology*, 137(1):75–85, 2012.

[47] Guenther Walther, Noah Zimmerman, Wayne Moore, David Parks, Stephen Meehan, Ilana Belitskaya, Jinhui Pan, and Leonore Herzenberg. Automatic clustering of flow cytometry data with density-based merging. *Advances in bioinformatics*, page 686759, 2009.

[48] J Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, February 2002.

[49] W.A. Moore and D.R. Parks. Update for the logicle data scale including operational code implementations. *Cytometry Part A*, 2012.

[50] A. Azad, J. Langguth, Y. Fang, A. Qi, and A. Pothen. Identifying rare cell populations in comparative flow cytometry. *Algorithms in Bioinformatics*, pages 162–175, 2010.

[51] A. Azad, S. Pyne, and A. Pothen. Matching phosphorylation response patterns of antigen-receptor-stimulated T cells via flow cytometry. *BMC Bioinformatics*, 13(Suppl 2):S10, 2012.

[52] H.T. Maecker, J.P. McCoy, and R. Nussenblatt. Standardizing immunophenotyping for the human immunology project. *Nature Reviews Immunology*, 2012.