HHS Public Access

Author manuscript

Cytometry A. Author manuscript; available in PMC 2024 November 01.

Published in final edited form as:

Cytometry A. 2023 November ; 103(11): 889–901. doi:10.1002/cyto.a.24776.

flowSim: Near Duplicate Detection for flow cytometry data

Sebastiano Montante[1], **Yixuan Chen**[1], **Ryan R. Brinkman**[1,2,*]

[1]Terry Fox Laboratory, BC Cancer Research, Vancouver, British Columbia, Canada.

[2]Department of Medical Genetics, University of British Columbia, Vancouver, British Columbia, Canada, 675 West 10th Avenue.

Abstract

The analysis of large amounts of data is important for the development of machine learning (ML) models. flowSim is the first algorithm designed to visualize, detect and remove highly redundant information in flow cytometry (FCM) training sets to decrease the computational time for training and increase the performance of ML algorithms by reducing overfitting. flowSim performs near duplicate image detection (NDD) by combining community detection algorithms with the density analysis of the marker expression values. flowSim clustering compared to consensus manual clustering on a dataset composed of 160 images of bivariate FCM data had a mean Adjusted Rand Index (ARI) of 0.90, demonstrating its efficiency in identifying similar patterns. flowSim selectively discarded near duplicate files in datasets constructed with known redundancy, and removed 92.6% of FCM images in a dataset of over 500,000 drawn from public repositories.

1 Introduction

Flow cytometry (FCM) technology is widely employed in immunology, including cancer diagnosis or treatment, and vaccine development to characterize and quantify the variety of cell types within fluid samples by measuring their biomarker expression levels [1, 2, 3, 4, 5, 6, 7]. The identification of interesting cell populations within each sample is called “gating” and it is the most critical step of FCM data analysis [2, 8, 9, 10]. The hierarchies and markers to analyze are pre-determined for each study driven by biological knowledge. Manual gating is both time consuming and very subjective, representing the principal source of variation in the application of the technology [11, 12, 13]. Even when the analysis is conducted by experts, personal biases result in inconsistencies within and across individuals, centers and time. The analysis of the same data performed by multiple operators across different centers can result in a coefficient of variation (CV) ranging from 17% to 44%, even with a standardized preparation of the samples [11,13]. To address these limitations,

*To whom correspondence should be addressed rbrinkman@bccrc.ca. Phone: 604-675-8132.

Author contributions

Sebastiano Montante: developer of the flowSim tool. Supervision of the performance tests and data generation for the performance evaluation.

Yixuan Chen: conducted the performance tests. Generation of main and supplementary images.

Ryan R. Brinkman: Conceived and reviewed the manuscript.

Code availability

The flowSim R package is freely available under the GPL license along with documentation at https://github.com/semontante/flowSim.

over 38 automated approaches, including machine learning approaches, have been developed to date [14, 15, 16, 17]. However, these all have limitations in either performance or parameterization time and manual analysis remains widely used [14, 15, 17, 18]. There is good availability of large amounts of raw data from the public repositories FlowRepository [19] and ImmPort [20], that can be used as the source of data for the development of ML approaches. NDD approaches with simple evaluation methods play an important role in decreasing the computation time for developing ML models (by reducing the size of the training dataset) and increasing performance (by reducing overfitting) [21, 22]. NDD is applied in many different fields such as genomics, languages, music and images [23]. Each application domain generates data that requires different considerations. While there have been attempts to universalize the features extraction process [24, 23], each data type has different features best suited to find similar images. There is an unmet need for a flexible NDD framework that can address the subjectivity of the real-world applications related to FCM research.

2 Methods

flowSim algorithm

Performing NDD in the context of FCM data requires the identification of bivariate marker expression images (i.e., standard flow cytometry plots) that depict the same objects (i.e., cell populations). The flowSim algorithm is divided in 3 main steps (Supplementary Figure 1) for two versions (V1, V2) of the algorithm with differing speed and accuracy:

1. Import the marker expression values and the density features of each input file.
2. Cluster the files based on the similarity distances:
    a. Generate the pairwise distance matrix, influenced by the tolerance parameter and the fragmentation parameter (V2 only).
    b. Convert the distance matrix into a network of nodes (each node is a file).
    c. Apply a community detection algorithm (Louvain algorithm or Infomap algorithm) on the network of nodes to find the clusters based on the number of connections among the nodes and the similarity scores (the connection weight).
3. Select the files to keep (influenced by the filtering parameter):
    a. Apply agglomerative hierarchical clustering on each cluster to identify the possible presence of subclusters (each subcluster contains files that have more similar density patterns than the rest of the cluster).
    b. Randomly select one file to keep from each subcluster of each cluster (or from the whole cluster if no subcluster is present).

Importing of the input files and density features

The input to flowSim is the path to a directory containing the files to analyze. Each file contains the expression values of two markers in comma separated value (csv) format. To improve efficiency, a common NDD approach is to extract feature vectors or more lightweight signatures/fingerprints [25] from documents to perform similarity matching. flowSim imports the expression values and the density estimates considering a grid of 50 points. Based on the density estimates, flowSim generates a set of 6 density features for each marker (for a total of 12 density features): mean of expression values, number of peaks, minimum and maximum peak position in the x axis, minimum and maximum peak height in the y axis. The peaks are defined as the set of density local maximums (y axis) of the expression values (x axis). The density estimates are calculated using the density function of the stats R package [26]. The density features are extracted using the findpeaks function of the pracma R package [27]. This function requires the vector of density estimates as input. It automatically identifies the number of peaks and provides the index of their position within the vector. The function also provides the height of each peak. The density estimates are used to calculate the similarity distances between two files (see below). The density features are used to identify the general density patterns in large datasets during the V2 mode of the flowSim algorithm and to perform the final filtering step (see below). A FCS to csv conversion function is available within the flowSim R package, based on the flowCore R package [28]. In this case the users need to select the pair of markers to consider. The users extract the bivariate data from a predefined gating hierarchy based on the aims of the future training (see Discussion).

Clustering of the files based on the similarity distances

The similarity-based clustering step includes two different modes that are chosen by the user-based on the number of files to process and machine memory. The first mode (V1) is the most accurate version of the algorithm but also the slowest and most memory intensive version. It is designed to work with small input datasets (less than 500 files) that require a relatively small amount of memory (less than 1 GB). The second mode (V2) is less accurate than V1 but is faster and more memory efficient. Files can be analyzed in parallel (see below).

V1 mode is divided into distance matrix generation and network clustering steps. During the distance matrix generation step (Supplementary Figure 2), flowSim calculates a similarity score for each pairwise combination of the input files applying a permutation test of equality using the “sm” R package [29]. The similarity score is defined as:

$$s\left(AB\right) = \frac{p1 + p2}{2} \quad (1)$$

Where:

s(AB) = similarity score between file A and file B. s(AB) ranges between 0 and 1. Higher values indicate a higher probability that the observed difference between the two densities is

due to chance (the difference is likely not statistically significant), with 1 indicating 100% probability. Values close to 0 indicate that the observed difference is unlikely to be due to chance.

p1 = p-value of the permutation test of equality for marker 1 between file A and file B.

p2 = p-value of the permutation test of equality for marker 2 between file A and file B.

In particular, for each marker, indicating with t the test statistic:

$$t = \sum_{i=1}^{N} (d_{1i} - d_{2i})^2 \quad (2)$$

Assuming we are calculating p1:

d_{1i} = ith density estimate of file 1 for marker 1.

d_{2i} = ith density estimate of file 2 for marker 1.

N = Size of the density estimate vector (N=50 for both files).

Considering each permutation as a single random data arrangement between the density estimates:

M = total number of mth random data arrangements between d_1 and d_2 (i.e., total number of permutations). Default value is 50. See flowMagic package documentation for additional details.

t_m = test statistic of the mth data arrangement.

t_0 = test statistic of the original data arrangement.

With p = p-value:

p1 = proportion of t_m as extreme as t_0 out of M.

If d_1 and d_2 are identical: $t_m = t_0$ for all mth arrangements.

Thus: p1=1 (the same process is applied for p2).

From the distance matrix of the input files, a network of nodes is generated. Each edge represents a similarity connection between two files. The nodes are clustered using a community detection algorithm based on the similarity scores (weight of each connection) and the number of connections (Supplementary Figure 3). The Louvain algorithm from the igraph R package [30] is the default community detection algorithm. flowSim allows the user to regulate the tolerance of differences within each cluster by filtering the similarity scores of the distance matrix (i.e., what range of similarity scores to consider during the network clustering). The tolerance parameter ranges from 0 to 1. For example, a parameter set to 0.4 means that all the similarity scores less than 0.4 will not be part of the network

generation and clustering step. In other words, a higher threshold means a more stringent similarity condition for the clustering of the files (differences are less tolerated within each cluster), while a lower score indicates a more flexible clustering of the files (differences are more tolerated within each cluster). The default value of the tolerance parameter is set to 0.6 for the V1 mode and 0.9 for the V2 mode, based on empirical testing (for V1 mode) and assumptions (for V2 mode, see supplementary file 1 for details). In summary, the parameter tuning is subjective based on the user's needs and the composition of each dataset. Suitable values tend to range between 0.6 (homogeneous/small datasets) and 0.9 (heterogeneous/large datasets).

Based on the network clustering, flowSim evaluates the heterogeneity of the dataset. The modularity score (0–1) measures the strength of the division of the network into modules (Supplementary Figure 3) and is calculated using the modularity function of the igraph R package. A higher modularity score means higher heterogeneity (more modules) while a lower score indicates a higher homogeneity (less modules). A heterogeneous dataset is more divided, with more small unconnected clusters (Supplementary Figure 3.A and and 3.B) than a homogeneous dataset (Supplementary Figure 3.C and 3.D) with many similar files. Modularity can represent an efficient measure of heterogeneity, however the modularity formula depends on the number of edges within and between each cluster. In very heterogeneous datasets the modularity score becomes less reliable due to the low number of edges and isolated files as it ignores the presence of isolated files that are important for the evaluation of the heterogeneity of a dataset. To address this, flowSim also considers the ratio of the number of isolated files to the number of clusters composed of at least 2 files. The final heterogeneity score (0–1) is the mean between the modularity score and the isolated files proportion (Supplementary Figure 4), with 1 indicating perfect heterogeneity.

V2 mode is divided into fragmentation, distance matrix generation for each part and network clustering. Generating a distance matrix from each pairwise combination of the input files becomes a time-consuming and memory intensive step as the size of the input dataset increases. V2 mode assumes that it is possible to approximate the calculation of all pairwise combinations described for the V1 mode (see supplementary file 1). It uses the density features generated during the import step to divide a large dataset into smaller batches that share the same general density patterns without the need to calculate the distances of each combination. This fragmentation step is performed by applying t-SNE [31] dimensionality reduction on the multidimensional density features set and then generating a low dimensional dataset by k-means clustering. The k parameter (number of clusters) of the k-means algorithm determines the number of batches in which the original dataset is divided. If the number of input files is less or equal to 2,000 files the k parameter is automatically estimated using the elbow method using the nClust function of the "anocva" R package [32]. If the dataset has more than 2,000 files the parameter needs to be manually set due to the high memory load and time of execution required by the elbow method (see supplementary file 1 for details). Each batch is then analyzed separately in the same way as in V1 mode. A distance matrix is generated for each batch and, eventually, all the resulting distance matrices are combined. The community detection algorithm is then applied on the combined distance matrix. Each batch represents a smaller space of analysis compared to the

original datasets and requires less time and less memory to be analyzed. As the number of batches increases the fragmentation of the dataset increases and the algorithm becomes less accurate, faster and requires less memory.

Filtering step

Regardless of the mode executed, once the files are clustered according to the selected community detection algorithm, each cluster is further analyzed by flowSim to select the files to keep. Each cluster may contain subclusters of expression patterns that may be worth keeping in the final dataset. For example, the same expression pattern may contain examples with a different density of the events (Supplementary Figure 1), which may influence the training of ML algorithms. The agglomerative hierarchical clustering algorithm is applied on each cluster using the set of features generated during importing. One random file is extracted from each subcluster found by the hierarchical clustering. With a lower threshold the algorithm keeps more patterns, while with a higher threshold it keeps less patterns. The value of 2.1 was set empirically under the assumption that the users want to keep 2–3 variations of the expression patterns (see Supplementary Figure 1). It ranges between 0 and an upper bound that depends on the heterogeneity (e.g., 0–3 in the datasets tested) within the cluster (the maximum distance value will be higher with more striking differences).

Visualization options

flowSim includes several visualization options generated using the visNetwork R package [33] to facilitate the exploration of the similarity relationships between the files and the clusters of the input dataset. The flowSim visualization framework helps the users to explore the heterogeneity and homogeneity of the datasets. The main element of the visualization framework is the network of nodes with 3 types of visualization (Supplementary Figure 5): complete network, partial network, contracted network. The default type is the complete network visualization.

Performance evaluation

Datasets

We first generated a set of 10 small datasets (160 files) to test the ability of flowSim to identify homogeneous patterns (data available from the flowSim GitHub page). From the original set of data containing thousands of samples, 15–25 files (FCS format) for each dataset (each containing the expression data of the same combination of markers) were selected to include different variations of the expression patterns, and converted them in a bivariate csv format using the flowCore R package. Dataset 2–10 included flow cytometry data collected from a prospective longitudinal infant cohort in The Gambia (West Africa) studied as part of the Human Immunology Project Consortium (HIPC) project [34]. Data were QC'd by Dr. Rym Ben Othman and QA'd by Dr. Arce Joann in partnership with the Precision Vaccines Program Data Management & Analysis Core. Dataset 1 included flow cytometry data (ONE Study project data) designed for the standardization of protocols across different multi-center biomarker studies [35]. flowSim similarity identification ability in both complex and simple situations was tested on datasets composed of images with

different types of feature differences such as the same number of peaks but with slightly different heights (subtle differences), different number of peaks, same number of peaks but where either position differed slightly or significantly or the mean of the expression values slightly changes or significantly. In general, the number of peaks represented the main discriminating factor, as it is easy to discern visually and it is usually biologically relevant (e.g., a different number of peaks usually means a different number of cell populations). Other features as discriminating factors were considered in order of relevance: position of the peaks, heights of the peaks, mean of expression.

34 large public datasets (559,819 files) were imported from flowRepository [19] and ImmPort [20] (see "supplementary file 2" in Supplementary Data for details). The data was compensated using the provided compensation matrix and transformed with the logicle transformation using the estimateLogicle function from the flowWorkspace R package [36].

A small "co-mixed dataset" was generated by the combination of different files from the original datasets. Each original file was mixed at progressively different percentages with the other original files to generate new mixed files progressively more different from the original files (see Supplementary Data) to finely control the balance of heterogeneity and homogeneity as well as the size of the data to analyze. Four files were selected from the small datasets and mixed to generate a small dataset of 50 mixed files. Large co-mixed datasets were also generated by co-mixing 4 files in different combinations at known percentages to generate a different number of bins (see Supplementary Data). Each bin represented a different pattern and contained near duplicate files (2 files randomly mixed many different times). Three co-mixed datasets composed of 13 bins (1,873 files), 31 bins (3,255 files) and 61 bins (6,576 files) were generated considering respectively a 25%, 10% and 5% of mixing.

Default tolerance value evaluation

The default tolerance value for the V1 mode was estimated by performing several test simulations on the 10 small datasets considering different thresholds for the tolerance parameter (from 0 to 1, by an increasing step of 0.1, for a total of 10 simulations for each dataset) with a manual reference (see "supplementary file 3"). This manual reference was designed for the tuning process by a researcher independently from the manual references used for the performance evaluation. For each dataset tested the range of tolerance values that gave the best ARI were determined.

Evaluation

The performance of flowSim results (in terms of pattern identification) was evaluated through comparison with five different manual references for each of the 10 small datasets (Reference A-E). To account for subjectivity of manual analysis each researcher generated a manual dendrogram for each dataset which depicted a hierarchical grouping sequence of the files, from the smallest groups on the bottom of the hierarchy (groups containing the most similar files) through the gradually larger groups while moving up the hierarchy (groups gradually containing less similar files) until reaching the top of the hierarchy (always the group that contains all files). A consensus cut line for all five hierarchies was

set, considering the cut lines that give the most similar results across all five references, to generate the final manual references for each dataset (see “supplementary file 4” in Supplementary Data). The final flowSim results generated using the default 0.6 value were compared with the results generated by fine tuning the tolerance parameter based on the composition and manual reference. The univariate density identity test from the “sm” R package was compared with the Jensen–Shannon divergence algorithm [37] applied to a bivariate density estimation. Accuracy was measured using the Adjusted Rand Index (ARI) [38]. The small datasets were also combined together and the relation between the clusters found by flowSim and the source dataset of the files (i.e., the original dataset they come from) was qualitatively analyzed. The default tolerance value and 0.9 in V1 mode was tested. The relation between the co-mixed files and original files of the small co-mixed dataset was also qualitatively analyzed.

The efficiency of the filtering was tested on large datasets using the V2 mode. Due to their high number of different density patterns, each cluster tended to be very large and included many subclusters. The 0.9 parameter value is stricter (differences are less tolerated) and was set as default for V2 under the assumption that most users would not prefer big differences within each cluster, as tends to happen in large datasets, but rather smaller homogeneous clusters (also easier to explore using the flowSim visualization framework). Each large dataset was analyzed and processed through several rounds of flowSim using the V2 mode, each reducing the residual homogeneity of the dataset. A new cycle was repeated until the percentage of files removed sharply decreased (by approximately 3.3 times, by manual inspection of the change), indicating that the set had become sufficiently heterogeneous and thus no more executions were necessary. The visualization framework and the heterogeneity score measured by flowSim helped to determine the final cycle, which showed a heterogeneity score of 0.9 and 90% of isolated nodes in the visualization framework. The public datasets were processed in parallel within a cluster of five machines using Slurm as cluster management system and job scheduler. Each computer node processed one dataset, with a mean of 5–10 datasets processed each day to complete one cycle using 4 cores (128 GB RAM, Intel Core i7 Processor). For the two large co-mixed datasets the number of files remaining for each bin at the end of each cycle was recorded. In ideal situations, the near duplicate files must be removed but the patterns must be preserved (i.e., at least one file needs to remain for each bin in the final cycle). In order to allow a measure of comparison of flowSim with possible future NDD tools, the number of files in overrepresented bins and number of total bins across all runs of flowSim for each dataset was recorded. We defined the overrepresented bins as the bins with a number of files greater than the mean bin size: 146 files for 13 bins, 105 files for the 31 bins and 108 for the 61 bins dataset.

The speed and memory load were evaluated on the large public datasets, comparing V1 and V2 modes (100 to 1,000 files in increments of 100). The evaluation was stopped at 1,000 files when the execution of V1 required about 16 hours with parallelized processing.

3 Results

Identification of homogeneous patterns

The first approach (analysis of 10 small datasets separately) compares the boxplots of the ARI values, calculated using the default value of the tolerance parameter, considering five manual references of homogeneous patterns. The mean ARI across all datasets and manual references is 0.86 (Figure 1, Supplementary Table 1). After fine tuning the tolerance parameter based on the datasets composition and manual references the average ARI was 0.90 (see “supplementary file 5” in Supplementary Data for additional results). The Jensen-Shannon Divergence algorithm showed inferior ARI values for all manual references, and the speed of execution was ten times slower (results not shown). The 0.6 default parameter value gave the best results in 80% of the datasets (Supplementary Table 2) considering a separate manual reference dedicated to the parameter tuning.

Results from combining the small datasets indicated that flowSim tended to cluster together files coming from the same datasets (Figure 2). The files were also co-mixed at different levels. As the level of co-mixing increased, the co-mixed files tended to have progressively less connections with the original files as shown in the network visualization and as was expected (Figure 3).

Efficiency of filtering

The execution of flowSim on 34 large public datasets reveals that the percentage of files removed per run (with each run of flowSim executed on the files filtered by the previous run) decreased at the third to fourth cycle (see Methods sections and “supplementary file 6” in Supplementary Data for the details of each cycle). As the datasets become progressively more heterogeneous the percentage of removed files decreases along the cycles (Supplementary Figure 6). Cycle 1 removed 73.8 % of files while the other cycles removed progressively a lower amount of files (46% in cycle 2, 32.6% in cycle 3, 22.4% in cycle 4). 559,819 images were reduced to 41,272 images in cycle 4. The processing of the 34 datasets required about 2 weeks. The time of execution and the memory load depended on the size of the datasets. The smaller datasets were processed in a few minutes requiring a few MB of memory load while the larger datasets required up to 1 hour with about 1 GB of memory load. For example, 1000 files were processed in about 10 minutes, requiring about 100 MB of memory. The first cycle was the slowest and the most memory intensive, the rest of the cycles were faster and required less memory as the number of files to process was less.

Large co-mixed datasets were also generated by co-mixing 4 files in different combinations at known percentages. The flowSim filtering and random filtering on the 13 bins co-mixed dataset were compared (Supplementary Figure 7). flowSim filtering discarded the near duplicate files within each bin (decreasing percentage of remaining files) while preserving all bins (at least 1 file remains in each bin). On the other hand, the random filtering discarded many bins (black regions). The tests on the 31 bins co-mixed dataset (Figure 4) showed similar results. The tests were performed using the default value of the filtering parameter (2.1). By decreasing the filtering threshold (1.2) it is also possible to preserve

the few bins removed in the 31 bins, however all bins in this situation become very similar (see Discussion). Figure 5 shows that the percentage of files in the overrepresented bins decrease across all runs of flowSim for all datasets tested. The percentage of total bins is constant (100%) across all runs for the 13 bins dataset (25% of mixing), 87% in the last run for the 31 bins dataset (10% of mixing) and 55.7% in the last run for the 61 bins dataset (5% of mixing). As the percentage of mixing decreases (and number of bins increases) the bins become more similar, thus the preservation of all bins becomes challenging and also questionable.

Speed and memory load

The technical performance of the two modes of flowSim execution was evaluated (see Methods section for the details of the two modes). The V1 mode showed higher time of execution and memory allocation compared to the V2 mode when using four cores (Figure 6). At 1,000 files V1 required about 16 hours and 1 GB of memory, while V2 required 10 minutes and 100 MB of memory. V2 is designed to work with larger datasets that require several tens of GB of memory in V1 mode (see supplementary file 1 for a detailed discussion of the V2 mode). In V2 mode, a machine with 1 GB of memory can analyze up to 10,000 files in 100 minutes.

The speed and memory efficiency tests using 1 core (Supplementary Figure 8) were stopped at 500 files, where the V1 mode required about 13 hours, about 4.5 times the time required by the 4 cores V1 tests at 500 files (2.9 hour). 500 files is also the suggested limit for the maximum number of files processable by the V1 mode in a reasonable amount of time (less than a half day on a typical desktop computer).

4 Discussion

No published research addresses the NDD task in FCM data. Most NDD papers describe NDD models or general principles of the NDD task, but not tools. There are no general reviews describing the application of NDD tools (only of general algorithms) and most algorithms [39, 23, 40, 41, 42, 43] refers to near-copy detection of fraudulent images, near-duplicate genetic sequences detection and other applications based on features hard to adapt in our context (i.e., they are features optimized to identify similar sequences of characters or digital copies). Based on carefully constructed datasets to aid in the unbiased comparative evaluation of the NDD task on FCM data our results indicate flowSim is fit for this purpose.

The small datasets were suitable to generate a manual reference of the homogeneous patterns, while the large datasets were suitable to test flowSim when there are thousands of patterns to filter. Our first tests evaluated flowSim based on a comparison with the manual groupings, suggesting that flowSim was fit for the NDD task and is tunable to a user's desired level of similarity given a dataset's composition. In addition, since files of the same dataset are assumed to be more similar than the files across datasets, flowSim clusters would then tend to cluster together files coming from the same datasets, and our results supported these assumptions. The application of flowSim on the small artificial co-mixed datasets of 50 files suggests that the tool is able to detect subtle differences among flow cytometry

images. The differences were purposely created to follow a gradual distribution from the original files that flowSim was able to detect. The evaluation of the scalability of flowSim using other 3 large co-mixed datasets revealed its applicability in very complex situations, requiring the analysis of the relationships of a high number of patterns at once and when the patterns are very similar. In this case, the discrimination between what to preserve and what to discard becomes more difficult. flowSim was able to preserve all patterns in the 13 bins dataset (at least 1 file in each bin survived the filtering) while removing only the redundant information. As such, this test was not sufficient to evaluate the efficiency of flowSim as it did not reach a failure point. Furthermore, real world large datasets may have more subtle differences among the patterns instead of clear divisions like in the 13 bins co-mixed dataset. The 31 bins and the 61 bins co-mixed dataset were designed to have less clear division among the patterns. This dataset was challenging for flowSim to analyze, as some “unique patterns” (bins) were so similar that they may be considered near-duplicates.

The very large size of the public dataset complicated the evaluation of flowSim as the removed plots would need to be manually reviewed and compared to the selected plots, a time consuming and subjective process. When applied to the combined 34 large public datasets of 559,819 files, flowSim removed 92.6% of the images after 4 iterations, implying that the overwhelming majority of bivariate plots associated with flow cytometry data images in the large publicly available datasets is redundant information for purposes of ML training. However, the determination of preservation of all patterns is challenging to demonstrate, and might not be desirable for all users’ definitions or what a related pattern actually is. The flexibility of flowSim allows users to personalize output by forcing flowSim to preserve more patterns (by decreasing the filtering threshold). The parameter tuning and the visualization framework are thus fundamental to its utility.

flowSim has a flexible visualization framework, adapting to small and bigger datasets, to homogeneous and heterogeneous datasets. For example, large heterogeneous datasets tend to be more complex to explore as they have a high number of different patterns to analyze. The contracted network visualization is useful in these cases, where the users can immediately identify the large interconnected clusters and the small clusters with no external connections. Users can switch to the partial network visualization mode to analyze only the disconnected clusters or they can focus their attention on the homogeneous part of the dataset to investigate such questions as whether homogeneity is evenly distributed or are there clusters of files more homogeneous than others. In the complete network visualization, the distance between two nodes (the length of the edges of the network) is proportional to the number of shared connections. For example, two nodes with only one connection and no shared connections with other nodes will have the largest distance (Supplementary Figure 9). This type of visualization shows the outliers of each cluster (Supplementary Figure 9.B) and the nodes that connect two clusters (Supplementary Figure 9.A and 9.C).

The main drawback of the flowSim algorithm is related to the univariate density space of its analysis. Literature on the NDD task is focused on finding the best features for NDD detection [40, 23]. In order to calculate the similarity between two files, flowSim compares the univariate density of the markers. Some expression patterns may be identifiable only in a multivariate space while they remain hidden in a univariate or bivariate density space.

The density of the events is only one important feature of a FCM image. In addition to the density, other important features include the geometry of the gates (e.g., rectangular, triangular) or the level of the marker expression (positive, negative, or neutral marker expression). Challenging situations include the analysis of datasets composed only by rare populations (i.e., small populations with few sparse events) or the comparison of two images that are different by only few sparse events (these few events may also have an important biological meaning). In these cases, the density is not enough to explain the similarity or dissimilarity between the images. The geometry of the gates or the level of marker expression can help the identification of similar images. Furthermore, bivariate FCM images appearing very similar may be given by different combinations of markers representing a completely different cell population and the user may want to consider this aspect of the data. In general, the interpretation of the content of each image (e.g, small rare populations, sparse events, level of marker expression, combination of markers and associated gates) could be different along with an ideal level of similarity. The NDD task applied in a FCM context needs to assure accuracy and flexibility at the same time. In theory the best solution for the NDD task would be a tool that exploits a machine learning algorithm trained on the multivariate space. This tool could potentially be faster and less memory intensive than flowSim, since it would be without any parametrization step (no pairwise comparison step, no distance matrix). The analysis of the multivariate space can also be potentially more accurate (it is rich with information), but only if the flexibility is granted. However, this approach implies that an ideal level of similarity between the images is available (the training set would be the reference). The tool would have potentially less flexibility, while needing to perform a task that is highly subjective. The flowSim parameters tuning is, at the same time, an advantage (it allows flexibility), a disadvantage (more complex to use) and, maybe, a necessity. This flexibility, determined by the tuning of several parameters, is the strongest advantage of flowSim. The tolerance, fragmentation and filtering are the parameters that mostly influence the final balance between homogeneity and heterogeneity. It is also possible to choose the type of community detection algorithm, the Louvain algorithm [44] to identify general similarity patterns or the Infomap algorithm [45] to identify more subtle patterns (Supplementary Figure 10). Additional documentation related to the parameters tuning can be found in supplementary file 1.

flowSim requires two or three parameters to set, and future research is required to reduce this number. Recently, a universal and a one parameter similarity metric, the normalized compression distance or NCD, has been employed effectively in diverse applications [23], like the analysis of large web documents. However, it still requires the setting of a similarity threshold. In addition, the context-dependent interpretation of experimental biological data plays an important role in determining the ideal similarity, influenced by a level of subjectivity not present in other fields (e.g., two similar web images of cats still represent two cats while two similar FCM images may not represent the same information). While in some fields eliminating the parameter tuning may be possible, in FCM data it may not be due to the user-dependent and problem-dependent metadata that is difficult to integrate in the mathematical definition of a “universal” similarity distance. Even if it were possible to identify a universal feature to express similarity, the subjectivity of the data interpretation and the desired level of similarity to retain would remain.

We propose the three 3 co-mixed datasets as a gold reference for testing other flow cytometry NDD tools as they stress the algorithms with their complex setup. Assessing the performance of NDD is a challenge due to the subjectivity of the definition of near duplication. The co-mixed datasets have known near duplication variability and force the testing tools to address two aspects of NDD algorithms relevant to NDD in FCM data: the efficiency of removing only redundant information ("accuracy") and the ability to adapt to complex subjective situations ("flexibility"). To quantitatively evaluate the accuracy, the number/percentage of files in the overrepresented bins and the total number/percentage of bins can be used. An accurate NDD tool decreases the number of files in the overrepresented bins (i.e., it removes near duplicate images) keeping constant the total number of bins (unique patterns are not removed). By measuring the difference of these two variables between flowSim (minuend of the difference) and other NDD tools (subtrahend of the difference) for each run and for each co-mixed dataset several cases of possible performance differences (Supplementary Figure 11) can be identified. The sign of the difference indicates the direction of the performance (superior or inferior) while the absolute value of the difference indicates how much the two performances diverge. In particular, if the difference is 0 for each run the two tools have the same performance. In this case, the number of files of overrepresented bins and the number of total bins kept by flowSim is the same as the other tool. If the difference is negative (Supplementary Figure 11.B and P), the number of files in overrepresented bins kept by the other tool is greater than the number of files kept by flowSim (flowSim has better performance), the number of total bins kept by the other tool is greater than flowSim (flowSim has less performance). In the ideal performance case (Supplementary Figure 11.I), the improved tool must show a positive difference for the number of files of overrepresented bins (the removing of the redundant files needs to be performed in one run) and a negative difference for the number of total bins (which must remain constant). The mean difference across all runs gives a general indication of the performance divergence (as it tends to diverge from 0, the difference in performance increases). For the three co-mixed datasets, flowSim generated a similar decrease in the percentages of files of overrepresented bins (Figure 5). Despite the increasing difficulty of the NDD task in the lower percentage mixed datasets (where bins became more similar), flowSim correctly detected overrepresented bins as containing redundant information. In the 10% and 5% mixing datasets, some whole bins were also detected as redundant information (using a default tolerance value), a questionable result that can be interpreted as correct or wrong depending on how the researcher interprets the ideal level of similarity. In the worst case, flowSim can remove only the files of the underrepresented bins while preserving the files of overrepresented bins (smaller decrease as the level of mixing decreases) as the high similarity of the bins is interpreted as undesirable by the algorithm. In summary, future tools aiming to overcome flowSim need to show: a greater preservation of the bins in the lower mixing datasets, a sharper decrease of the files of overrepresented bins across all datasets and the flexibility in adapting to different situations. Flexibility is both a qualitative property (e.g., presence or absence of parameters to alter default filtering, presence of a visualization framework) and quantitative property (e.g., how many parameters? In general a low number of parameters is better) of the algorithm whose ideal output depends on the user's need and on the dataset composition as underlined across all the sections of this paper.

It is important to underline that flowSim focuses only on the pairs of markers provided by the users ignoring the gating hierarchy they originated from. The combination of markers analyzed by flowSim can be either the same pair of markers extracted from a single gating hierarchy or the different pairs of markers extracted from different predefined gating hierarchies. In other words, flowSim does not automatically consider all possible pairs of markers of an experiment, it considers the pairs of markers selected by the users. The level of information captured by the tool depends on the users. flowSim assumes that the training algorithm of the users works on the bivariate data related to a specific pair of markers or different pairs of markers. For example, the users may aim to develop a machine learning algorithm to gate specific populations of interest. The users will provide the bivariate data of the populations they want to pre-process with flowSim. A training algorithm aimed to identify the Singlets population will require files reporting FSC-H and FSC-A data. Since the identification of the Singlets using FSC-H and FSC-A is a common gating step in many gating hierarchies, the users can provide bivariate data (for this gating step) extracted from different gating sequences. A training algorithm aimed to identify a rare population defined only by a specific sequence of markers will require bivariate data related to that specific step of the gating sequence. The users may want to develop a training algorithm that can identify different populations extracted from different pairs of markers (independently from any hierarchy). In this case the users will provide the bivariate data related to these multiple populations.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgements

We thank the study participants and their families for providing the biosamples upon which this study is based, Daniel Yokosawa and Razzi Movassaghi for pre-processing the FlowRepository and ImmPort datasets, Tobias R. Kollman and the Expanded Program on Immunization Consortium (EPIC) for sharing flow cytometry data files of the Gambia main cohort as well as the Boston Children's Hospital Precision Vaccines Program Data Management & Analysis Core (DMAC) for assistance with QC, QA and deposition of de-identified data into ImmPort. We thank Justin Meskas for his suggestions during manuscript proofreading.

Funding

NSERC, NSERC-CREATE, NFRFR-2021-00300. The research reported in this publication was also supported by the National Institute Of Allergy And Infectious Diseases of the National Institutes of Health under Award Number U19AI118608. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Data availability

Large co-mixed datasets and small test datasets are freely available under the GPL license at https://github.com/semontante/flowSim_test_data. The original files used to generate the test data (Dataset 2–10, HIPC project) are freely available on ImmPort (study accession number: SDY1538). The original files for Dataset 1 (OneStudy project) are freely available on flowRepository (flowRepository ID: R-FCM-ZYQT).

References

1. Abraham RS and Aubert G. “Flow Cytometry, a Versatile Tool for Diagnosis and Monitoring of Primary Immunodeficiencies.” Clinical and vaccine immunology : CVI vol. 23,4 254–71. 4 Apr. 2016, doi:10.1128/CVI.00001-16
2. Bashashati A. and Brinkman RR “A survey of flow cytometry data analysis methods.” Advances in bioinformatics vol. 2009 (2009): 584603. doi:10.1155/2009/584603 [PubMed: 20049163]
3. De Rosa SC et al. “Vaccination in humans generates broad T cell cytokine responses.” Journal of immunology (Baltimore, Md. : 1950) vol. 173,9 (2004): 5372–80. doi:10.4049/jimmunol.173.9.5372
4. Lin L. et al. “COMPASS identifies T-cell subsets correlated with clinical outcomes.” Nature biotechnology vol. 33,6 (2015): 610–6. doi:10.1038/nbt.3187
5. Kalina T. et al. “EuroFlow standardization of flow cytometer instrument settings and immunophenotyping protocols.” Leukemia 26, 1986–2010 (2012). 10.1038/leu.2012.122 [PubMed: 22948490]
6. Macchia I. et al. “Immune monitoring in cancer vaccine clinical trials: critical issues of functional flow cytometry-based assays.” BioMed research international vol. 2013 (2013): 726239. doi:10.1155/2013/726239 [PubMed: 24195078]
7. Turtle CJ “Chimeric antigen receptor modified T cell therapy for B cell malignancies.” International journal of hematology vol. 99,2 (2014): 132–40. doi:10.1007/s12185-013-1490-x [PubMed: 24338745]
8. Rahim A. et al. “High throughput automated analysis of big flow cytometry data.” Methods (San Diego, Calif.) vol. 134-135 (2018): 164–176. doi:10.1016/j.ymeth.2017.12.015 [PubMed: 29287915]
9. Verschoor CP et al. “An Introduction to Automated Flow Cytometry Gating Tools and Their Implementation.” Frontiers in immunology vol. 6 380. 27 Jul. 2015, doi:10.3389/fimmu.2015.00380
10. McKinnon KM “Flow Cytometry: An Overview.” Current protocols in immunology vol. 120 5.1.1–5.1.11. 21 Feb. 2018, doi:10.1002/cpim.40
11. Grant R. et al. “Quantifying Operator Subjectivity within Flow Cytometry Data Analysis as a Source of Measurement Uncertainty and the Impact of Experience on Results.” PDA journal of pharmaceutical science and technology vol. 75,1 (2021): 33–47. doi:10.5731/pdajpst.2019.011213 [PubMed: 33067330]
12. Baumgaertner P. et al. “Unsupervised Analysis of Flow Cytometry Data in a Clinical Setting Captures Cell Diversity and Allows Population Discovery.” Frontiers in immunology vol. 12 633910. 30 Apr. 2021, doi:10.3389/fimmu.2021.633910
13. Finak G. et al. “Standardizing Flow Cytometry Immunophenotyping Analysis from the Human ImmunoPhenotyping Consortium.” Scientific Reports 6 (February): 20686 (2016). [PubMed: 26861911]
14. Cheung M. et al. “Current trends in flow cytometry automated data analysis software.” Cytometry. Part A : the journal of the International Society for Analytical Cytology vol. 99,10 (2021): 1007–1021. doi:10.1002/cyto.a.24320 [PubMed: 33606354]
15. Liu P. et al. “Recent Advances in Computer-Assisted Algorithms for Cell Subtype Identification of Cytometry Data.” Frontiers in cell and developmental biology vol. 8 234. 28 Apr. 2020, doi:10.3389/fcell.2020.00234
16. Hu Z. et al. “Application of Machine Learning for Cytometry Data.” Frontiers in immunology vol. 12 787574. 3 Jan. 2022, doi:10.3389/fimmu.2021.787574
17. Weber LM and Robinson MD “Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data.” Cytometry. Part A : the journal of the International Society for Analytical Cytology vol. 89,12 (2016): 1084–1096. doi:10.1002/cyto.a.23030 [PubMed: 27992111]
18. Lee H. et al. “High-Throughput Analysis of Clinical Flow Cytometry Data by Automated Gating.” Bioinformatics and biology insights vol. 13 1177932219838851. 3 Apr. 2019, doi:10.1177/1177932219838851

19. Spidlen J. et al. “FlowRepository: a resource of annotated flow cytometry datasets associated with peer-reviewed publications.” Cytometry. Part A : the journal of the International Society for Analytical Cytology vol. 81,9 (2012): 727–31. doi:10.1002/cyto.a.22106 [PubMed: 22887982]
20. Bhattacharya S. et al. “ImmPort: disseminating data to the public for the future of immunology.” Immunologic research vol. 58,2–3 (2014): 234–9. doi:10.1007/s12026-014-8516-1
21. Chum O. et al. “Scalable near Identical Image and Shot Detection.” In Proceedings of the 6th ACM International Conference on Image and Video Retrieval, 549–56. CIVR ‘07. New York, NY, USA: Association for Computing Machinery (2007).
22. Thyagharajan KK and Kalaiarasi G. “A Review on Near-Duplicate Detection of Images using Computer Vision Techniques.” Arch Computat Methods Eng 28, 897–916 (2021). 10.1007/s11831-020-09400-w
23. Xi Z. et al. “Effective and Fast Near Duplicate Detection via Signature-Based Compression Metrics”, Mathematical Problems in Engineering, vol. 2016, Article ID 3919043, 12 pages, 2016. 10.1155/2016/3919043
24. Cilibrasi R. and Vitanyi P. “Clustering by compression,” IEEE Transactions on Information Theory, vol. 51, no. 4, pp. 1523–1545, 2005.
25. Mitzenmacher M. et al. “Efficient estimation for high similarities using odd sketches” in Proceedings of the 23rd International Conference on World Wide Web (WWW ‘14), pp. 109–118, Florence, Italy, April 2014.
26. R Core Team. “R: A language and environment for statistical computing.” (2018) R Foundation for Statistical Computing,Vienna, Austria. URL https://www.R-project.org/.
27. Borchers HW “pracma: Practical Numerical Math Functions.” (2018) R package version 2.2.2. https://CRAN.R-project.org/package=pracma.
28. Ellis B. et al. “flowCore: Basic structures for flow cytometry data.” (2022) R package version 2.8.0.
29. Bowman AW and Azzalini A. R package ‘sm’: nonparametric smoothing methods (version 2.2–5.7) URL http://www.stats.gla.ac.uk/~adrian/sm (2021).
30. Csardi G. and Nepusz T. “The igraph software package for complex network research.” InterJournal, Complex Systems, 1695 (2006)
31. Krijthe JH “Rtsne: T-Distributed Stochastic Neighbor Embedding using Barnes-Hut Implementation” (2015) R package version 0.16
32. Fujita A. et al. “A non-parametric statistical test to compare clusters with applications in functional magnetic resonance imaging data.” Statistics in medicine vol. 33,28 (2014): 4949–62. doi:10.1002/sim.6292 [PubMed: 25185759]
33. Almende BV et al. “visNetwork: Network Visualization using ‘vis.js’ Library.” (2018) R package version 2.0.5.
34. Idoko OT et al. “Clinical Protocol for a Longitudinal Cohort Study Employing Systems Biology to Identify Markers of Vaccine Immunogenicity in Newborn Infants in The Gambia and Papua New Guinea.” Frontiers in pediatrics vol. 8 197. 30 Apr. 2020, doi:10.3389/fped.2020.00197
35. Ivison S. et al. “A standardized immune phenotyping and automated data analysis platform for multicenter biomarker studies.” JCI insight vol. 3,23 e121867. 6 Dec. 2018, doi:10.1172/jci.insight.121867
36. Finak G. and Jiang M. “flowWorkspace: Infrastructure for representing and interacting with gated and ungated cytometry data sets.” (2022) R package version 4.8.0.
37. Drost H. “Philentropy: Information Theory and Distance Quantification with R.” Journal of Open Source Software, 3(26), 765, 10.21105/joss.00765 (2018)
38. Santos JM and Embrechts M. “On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification.” Lecture Notes in Computer Science, vol 5769. Springer, Berlin, Heidelberg. 10.1007/978-3-642-04277-5_18 (2009).
39. Zhou Z. et al. “Fast and accurate near-duplicate image elimination for visual sensor networks.” International Journal of Distributed Sensor Networks. 2017;13(2). doi:10.1177/1550147717694172
40. Qiao F. et al. “Large scale near-duplicate celebrity web images retrieval using visual and textual features.” TheScientificWorldJournal vol. 2013 795408. 14 Sep. 2013, doi:10.1155/2013/795408

41. Zhang Y. et al. “Single- and Cross-Modality Near Duplicate Image Pairs Detection via Spatial Transformer Comparing CNN.” Sensors (Basel, Switzerland) vol. 21,1 255. 2 Jan. 2021, doi:10.3390/s21010255

42. Kim H. et al. “BASIL: Effective Near-Duplicate Image Detection Using Gene Sequence Alignment.” In: Advances in Information Retrieval. ECIR 2010. Lecture Notes in Computer Science, vol 5993. Springer, Berlin, Heidelberg. 10.1007/978-3-642-12275-0_22.

43. Kwon Y. et al. “Identifying and removing duplicate records from systematic review searches.” Journal of the Medical Library Association : JMLA vol. 103,4 (2015): 184–8. doi:10.3163/1536-5050.103.4.004 [PubMed: 26512216]

44. Traag VA et al. “From Louvain to Leiden: guaranteeing well-connected communities” Sci Rep. 2019 Mar 26;9(1):5233. doi: 10.1038/s41598-019-41695-z. [PubMed: 30914743]

45. Tandon A, et al. “Community detection in networks using graph embeddings.” Phys Rev E. 2021 Feb;103(2–1):022316. doi: 10.1103/PhysRevE.103.022316. [PubMed: 33736102]

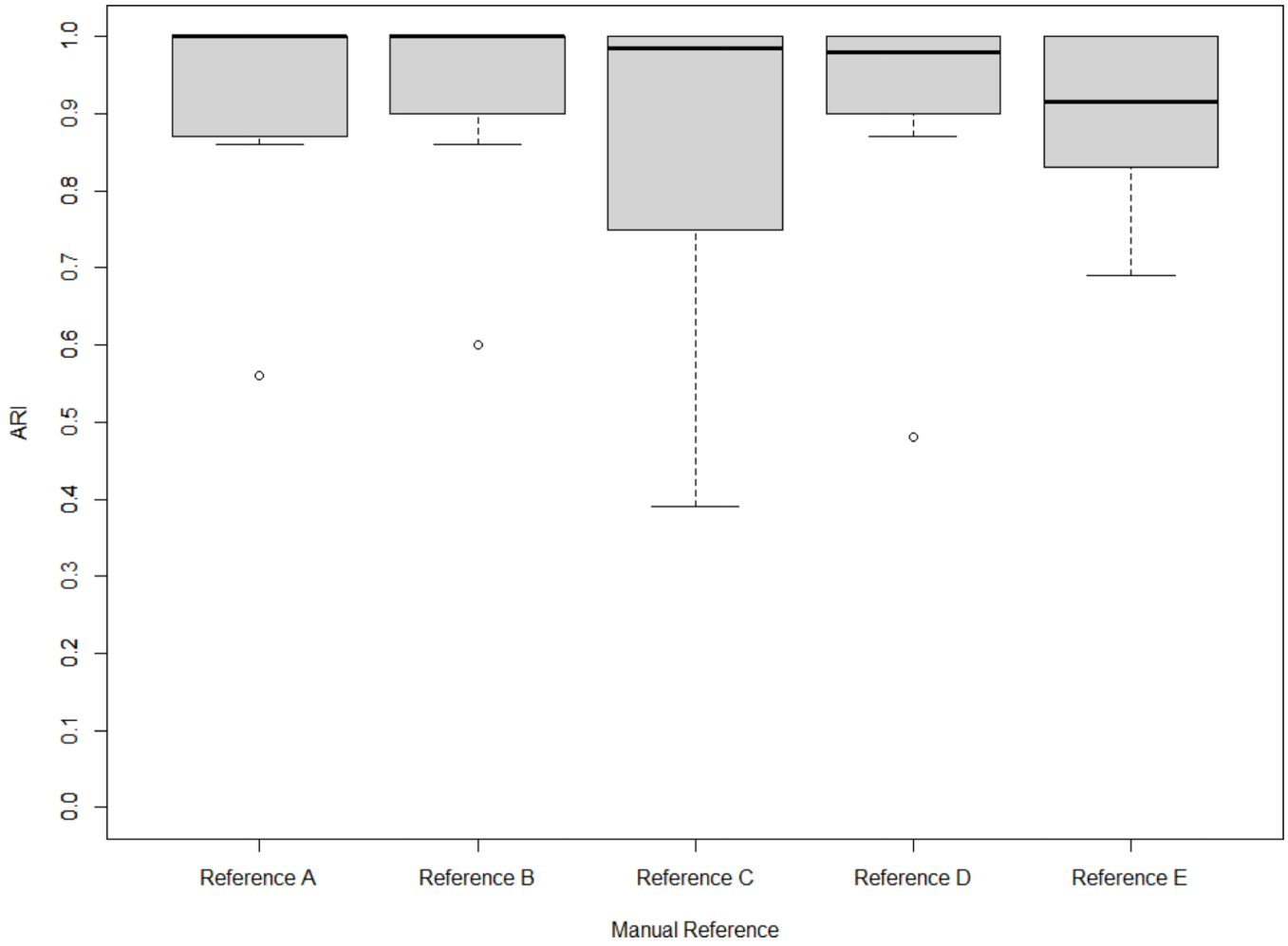

Figure 1:

flowSim accuracy on small datasets. Boxplots of ARI values calculated using the flowSim V1 mode (default tolerance) versus the 5 manual references.

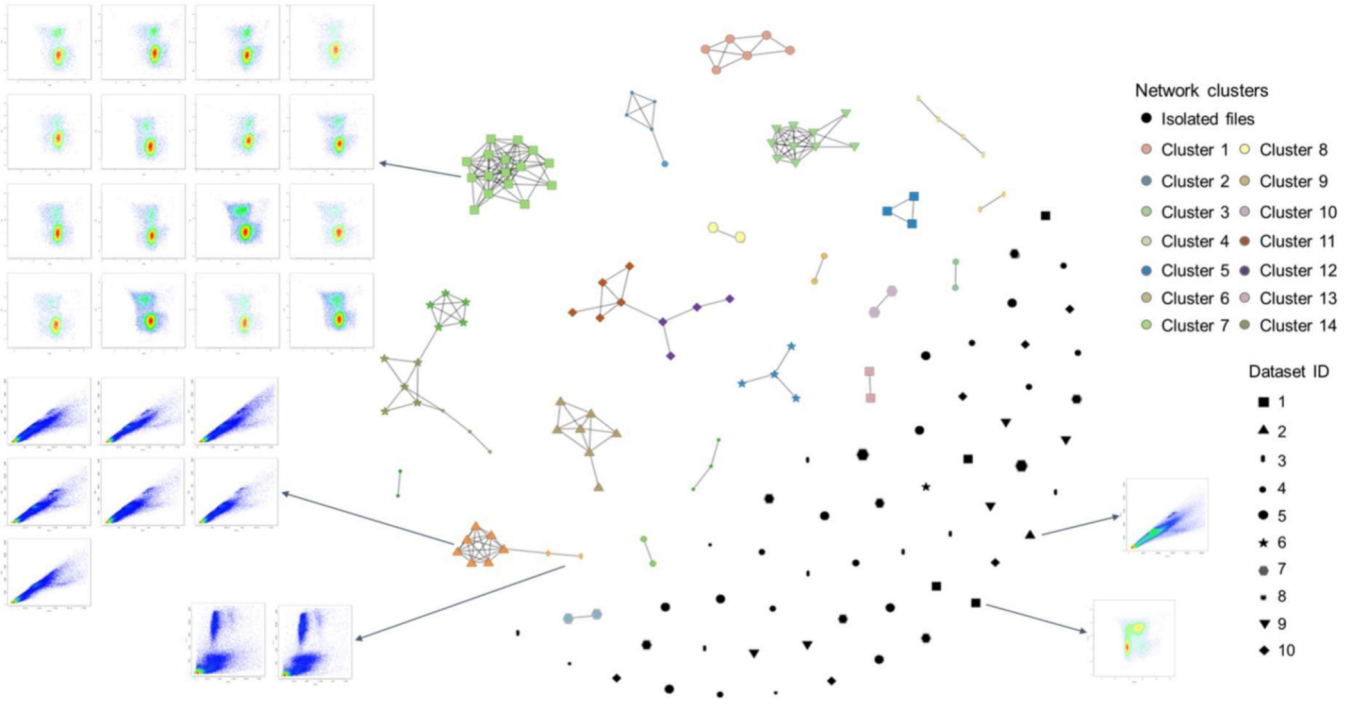

Figure 2:

flowSim homogeneous patterns analysis on the small datasets combined together (complete network visualization, 0.9 tolerance value). Example of homogeneous patterns identified by flowSim. Nodes with the same shape indicate files from the same dataset; same colour indicate files of the same flowSim cluster.

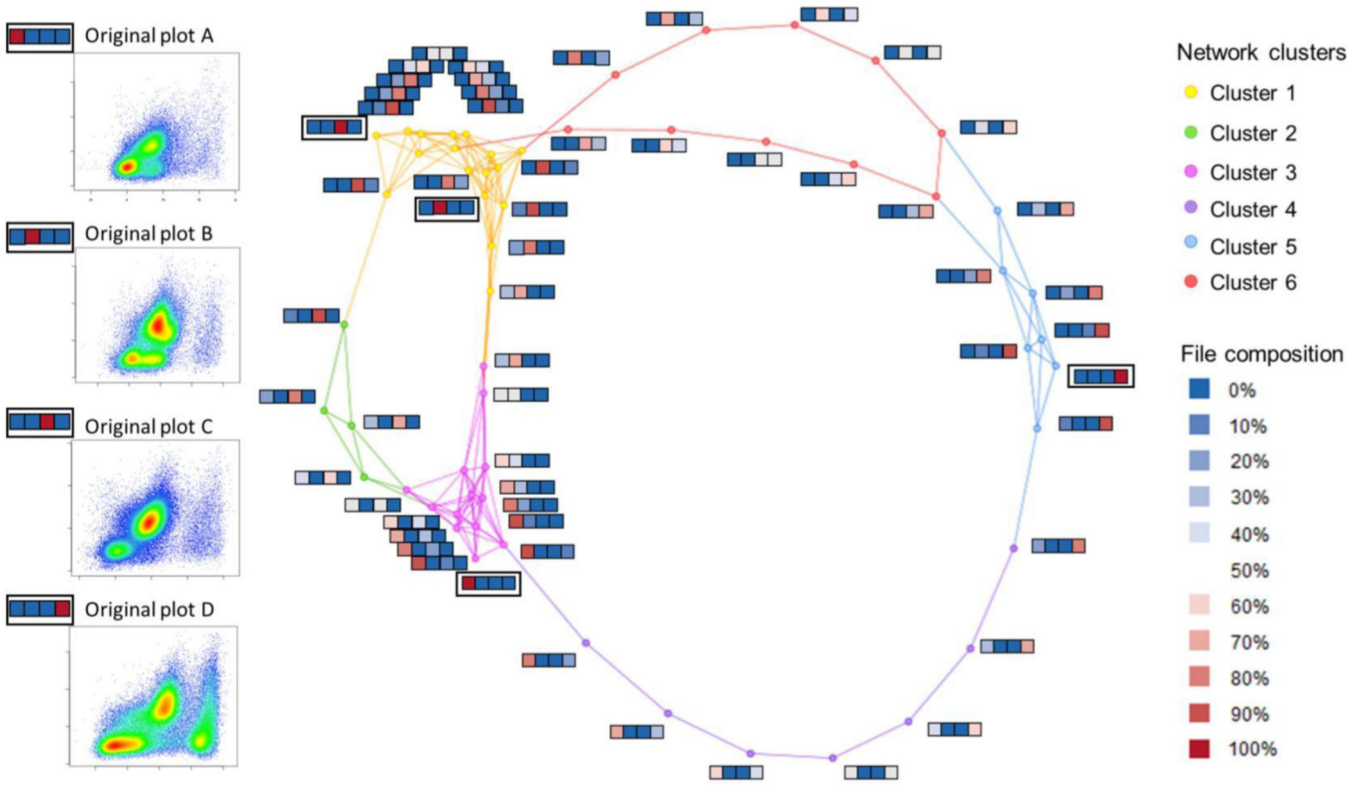

Figure 3:

flowSim homogeneous patterns analysis after co-mixing the files. Network of a co-mixed dataset with 4 original plots. Next to each node (mixed file) it is indicated the mixing composition, represented by 4 squares (4 original plots) coloured based on the percentage of original plots mixed in that file.

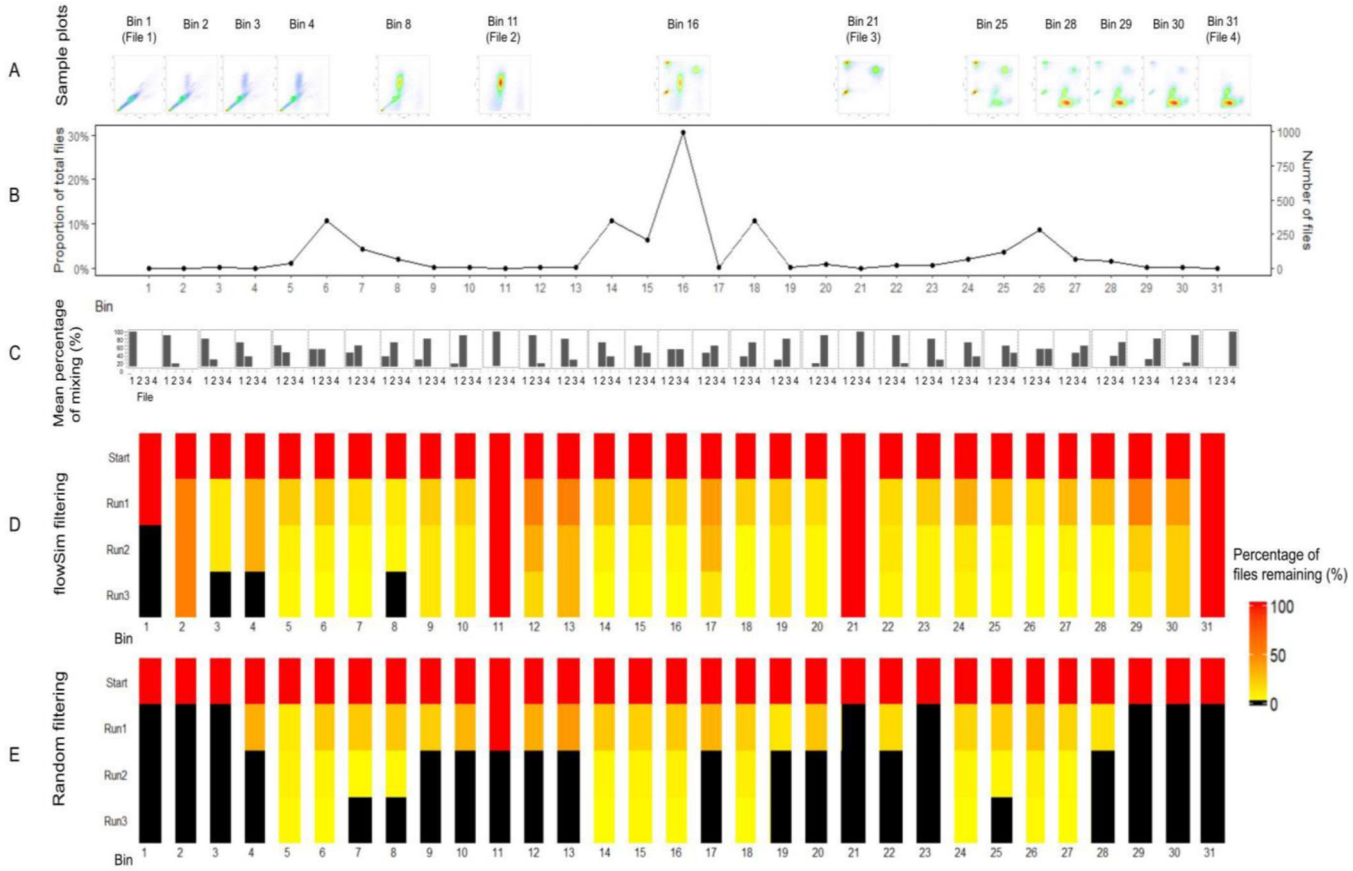

Figure 4:

flowSim filtering result and random filtering result for the 31 bins co-mixed dataset (10% mixture change between the bins). From top to bottom: examples of bins (A), line plots of percentages of total files in the y-axis and bins id in the x-axis (B), mean percentages of mixing between the files (C), percentage of files remaining in each bin for each run of flowSim filtering (D), percentage of files remaining in each bin for each run of random filtering (E).

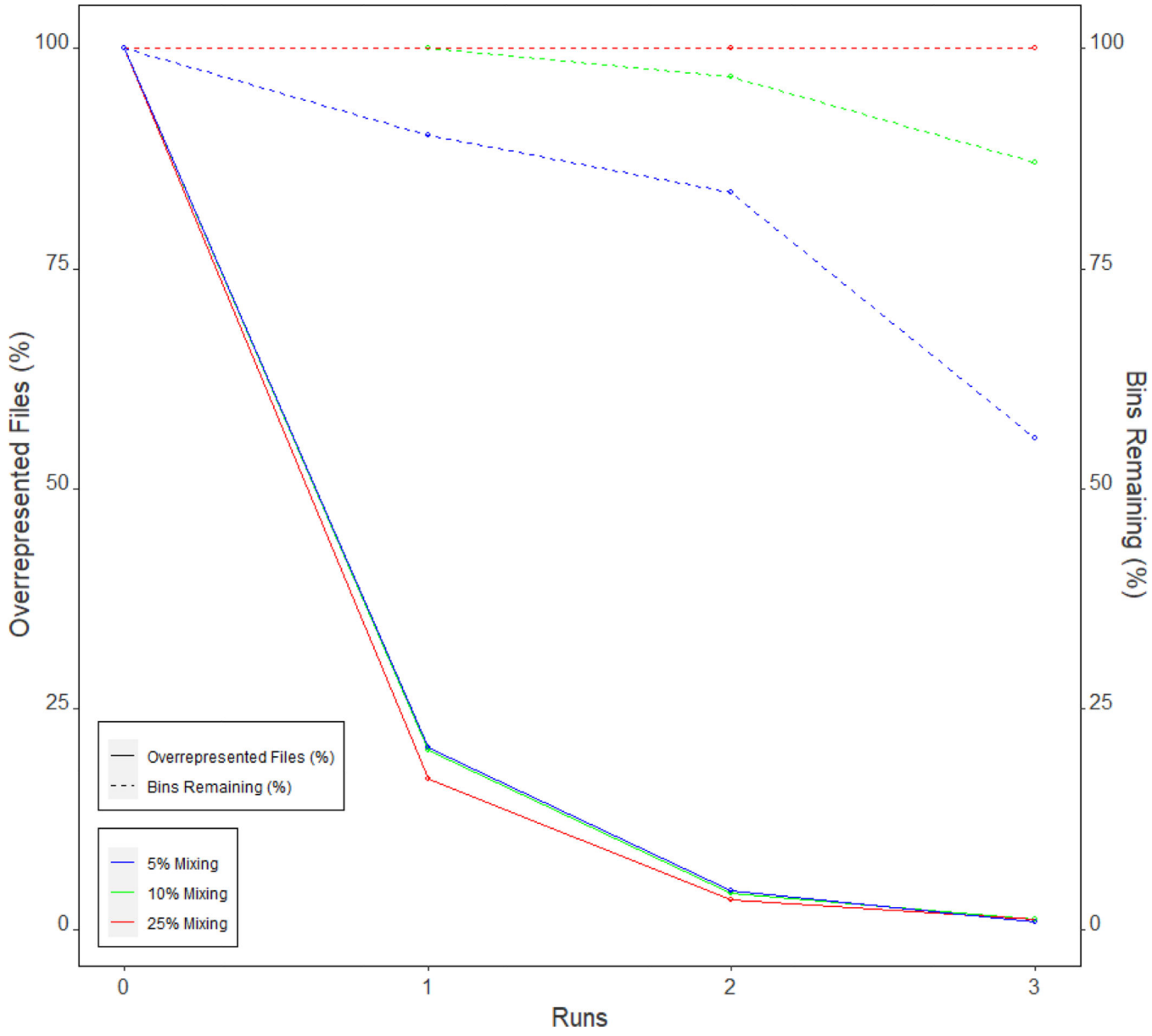

Figure 5:

Runs of flowSim vs percentages of files in overrepresented bins (continued line) and percentages of bins (dotted line). The different colours indicate a test for a different co-mixed dataset.

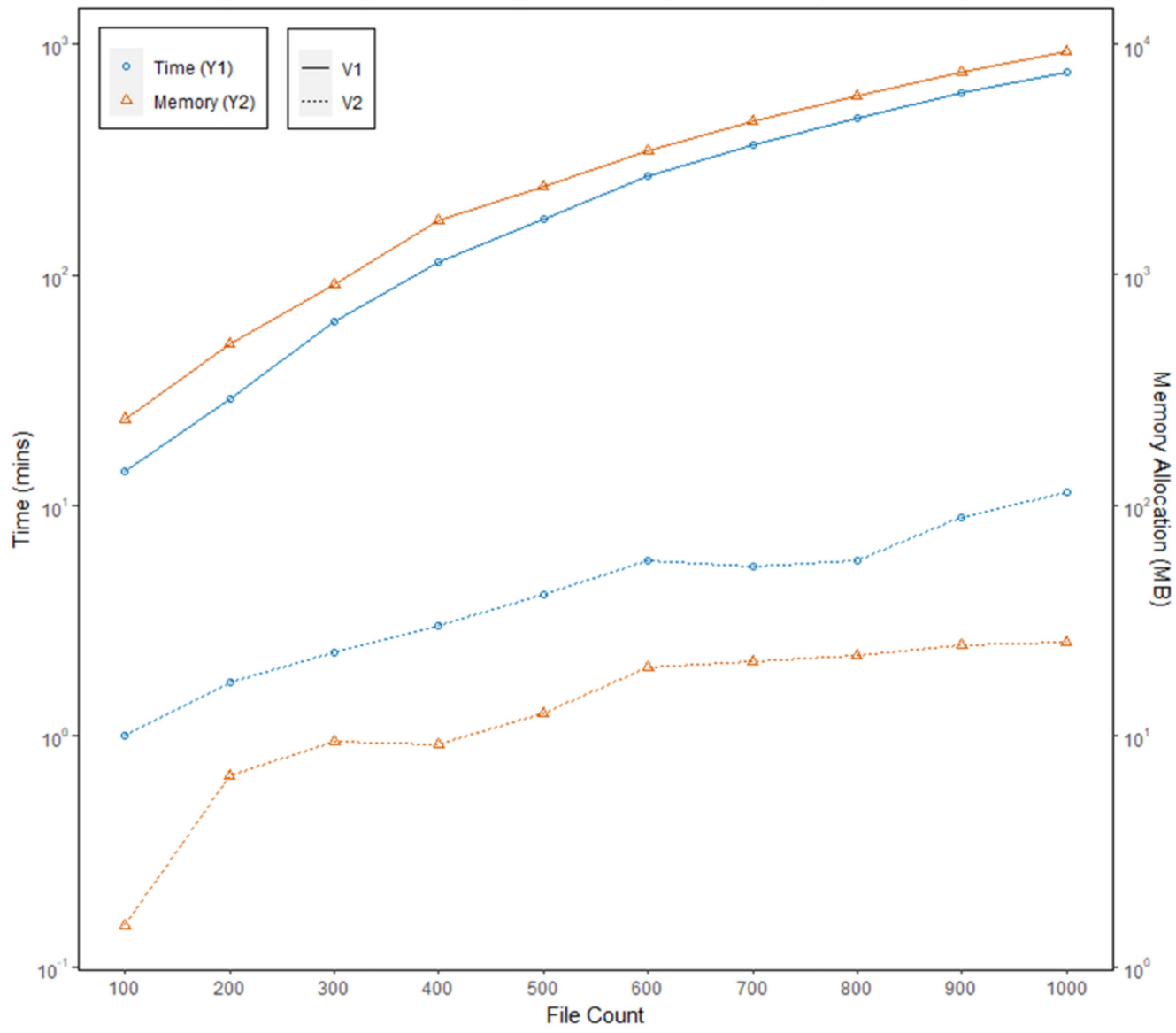

Figure 6:

Speed and memory tests (4 cores). As the number of files increases (x axis), the memory (triangles) and time of execution (circles) are compared using V1 (solid line) and V2 mode (dotted line).