Comparison of Clustering Methods for High-Dimensional Single-Cell Flow and Mass Cytometry Data

Additional results for clustering algorithm *densityCut*

Lukas M. Weber[1,2], Mark D. Robinson[1,2,*]

[1] Institute of Molecular Life Sciences, University of Zurich, Zurich, Switzerland

[2] SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland

[*] mark.robinson@imls.uzh.ch

February 15, 2017

1 Introduction

This report contains additional results extending the analyses in our paper comparing clustering methods for high-dimensional flow and mass cytometry (CyTOF) data, published in the journal *Cytometry Part A* (Weber and Robinson, 2016):

- Weber L. M. and Robinson M. D. (2016) *Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data.* Cytometry Part A, 89A: 1084–1096. Available at: http://onlinelibrary.wiley.com/doi/10.1002/cyto.a.23030/full.

The analyses in our paper were designed to be extensible, to allow ourselves and other researchers to include new clustering algorithms and reference data sets. The paper is accompanied by a code repository (GitHub: https://github.com/lmweber/cytometry-clustering-comparison) and a data repository (FlowRepository: https://flowrepository.org/id/FR-FCM-ZZPH), containing R scripts and pre-processed data files to reproduce all published analyses.

R scripts to reproduce the additional analyses in this report, along with a final copy of the report, are also available from the GitHub repository (https://github.com/lmweber/cytometry-clustering-comparison).

2 densityCut

Overview

This report evaluates the performance of the new clustering method `densityCut` (Ding et al., 2016). The `densityCut` algorithm is based on density estimation in K-nearest-neighbor graphs, with improved computational efficiency compared to previous similar approaches. Unlike many other methods, `densityCut` is designed for clustering a wide range of biological data types, including CyTOF data; instead of only a single data type.

In the paper introducing `densityCut` (Ding et al., 2016), the authors evaluate performance on a number of synthetic and experimental data sets. For the evaluations on CyTOF data, they use three data sets derived from the healthy bone marrow data sets in Levine et al. (2015) (which were also used to generate data sets `Levine_32dim` and `Levine_13dim` in our paper). Similar to our evaluations, the authors use manually gated cell population labels as the reference standard. Performance is compared against the `PhenoGraph` algorithm (Levine et al., 2015). However, there are several important differences in the evaluation methodology, which affect the results (see Discussion).

Implementation and availability

The `densityCut` clustering method is implemented as an R package, which is freely available for download from BitBucket (https://bitbucket.org/jerry00/densitycut_dev).

However, installation is non-trivial. The package is provided as a source package (`.tar.gz` file), and the README documentation in the BitBucket repository provides commands to install the source package from the command line. However, this is a difficult procedure for users who are unfamiliar with the command line. Installation also requires several dependency packages (listed in the DESCRIPTION file in the package source code), which need to be installed separately, or the installation will fail. In addition, installation of the source package and dependencies requires a compiler (e.g. `clang`), which may not be pre-installed on all systems (especially Windows systems); it is unlikely that novice users will be able to set up a compiler if required. R packages can also be installed directly from BitBucket using the `devtools` package (R commands: `library(devtools)`; `install_bitbucket("jerry00/densitycut_dev")`), which simplifies some of these steps (although a compiler is still required), but this is not mentioned in the documentation.

Example data sets from the evaluations in the original paper are provided within the R package, which is useful for new users. A short code example is provided in the README documentation in the BitBucket repository, and the main function help file includes an extended example. However, an extended workflow example (e.g. Bioconductor vignette) would be helpful for new users.

Parameters

There are two main parameters in the `densityCut` algorithm: K (number of neighbors in the K-nearest-neighbor graph), and *alpha* (damping factor). These are described in detail in the paper. Default values are provided, which are shown to give good results across a range of data settings.

The number of clusters is automatically determined by the algorithm. As discussed in our paper (Weber and Robinson, 2016), while this may be useful in many situations, it can create difficulties for analyzing high-dimensional cytometry data, since there is not necessarily any "true" number of cell populations (see Discussion in main paper). Similar to many other clustering methods, the final number of clusters can be adjusted by tuning the indirect parameters, but this is difficult.

3 Methods

We evaluated `densityCut` using the same data sets and same evaluation methodology as in our main paper (Weber and Robinson, 2016). For additional details, refer to the main paper.

Parameter settings

We used default values for all parameters required by `densityCut`: $K = log_2(N)$, where N is the number of cells; and $alpha = 0.90$.

We used the automatic number of clusters for the final results, since this gave reasonable results overall (the same strategy as described in the main paper). The number of clusters for each data set is summarized in Table 1, which extends Supporting Information Table S3 from the original paper.

Subsampling and multiple processor cores were not required, since the `densityCut` algorithm is relatively efficient. Subsampling and the number of processor cores are summarized in Table 2, which extends Supporting Information Table S4 from the original paper. We ran the evaluations on our Linux server (see main paper and Supporting Information Table S1).

All other methodological details were the same as described in the main paper and supporting information.

Table 1. Number of clusters.

Method	Data set						Selection options		
	Lev_32	Lev_13	Sam_01	Sam_all	Nil_rare	Mos_rare	Automatic	Indirect parameters	Manual
densityCut	11	10	13	19	25	15	✔*	✔	

Options available for selecting the number of clusters are indicated with check marks (✔). Star indicates the option used when multiple options were available (✔*). This table extends Supporting Information Table S3 from the original paper, which contains entries for all other methods.

Table 2. Subsampling and number of processor cores.

Method	No. of data points						No. of cores
	Lev_32	Lev_13	Sam_01	Sam_all	Nil_rare	Mos_rare	
densityCut	all	all	all	all	all	all	1

No subsampling was required for entries labeled "all". Hardware specifications are provided in Supporting Information Table S1 (spreadsheet file) from the original paper. This table extends Supporting Information Table S4 from the original paper, which contains entries for all other methods.

4 Results: Clustering method *densityCut*

Tables of results

Table 3. Results for clustering method *densityCut*.

	Multiple populations of interest								Single rare population of interest			
	Levine_32dim		Levine_13dim		Samusik_01		Samusik_all		Nilsson_rare		Mosmann_rare	
	mean F1	runtime hh:mm:ss	mean F1	runtime hh:mm:ss	mean F1	runtime hh:mm:ss	mean F1	runtime hh:mm:ss	F1	runtime hh:mm:ss	F1	runtime hh:mm:ss
densityCut	0.584	00:26:53	0.260	00:03:05	0.429	00:06:51	0.480	04:35:58	0.140	00:00:18	0.756	00:16:05

Results show the mean F1 score for data sets with multiple cell populations of interest, and F1 score for data sets with a single rare cell population of interest; as well as runtimes. Runtimes are not precisely comparable between methods due to differences in subsampling, number of processor cores, and hardware specifications (Supporting Information Tables S1 and S4 from the original paper); however they are included in order to provide users with information about order-of-magnitude differences. This table extends Table 3 from the original paper, which contains entries for all other methods.

Table 4. Results for FlowCAP data sets.

Method	Updated evaluation methodology		FlowCAP-I evaluation methodology	
	FlowCAP_ND	FlowCAP_WNV	FlowCAP_ND	FlowCAP_WNV
	mean F1	mean F1	mean F1	mean F1
densityCut	0.429	0.654	0.903	0.856

Results show the mean F1 score for each data set. Differences in evaluation methodologies are explained in Materials and Methods (main paper). Additional details are included in Supporting Information Methods, Data sets, from the original paper. This table extends Supporting Information Table S5 from the original paper, which contains entries for all other methods.

Figures: Main results for all data sets

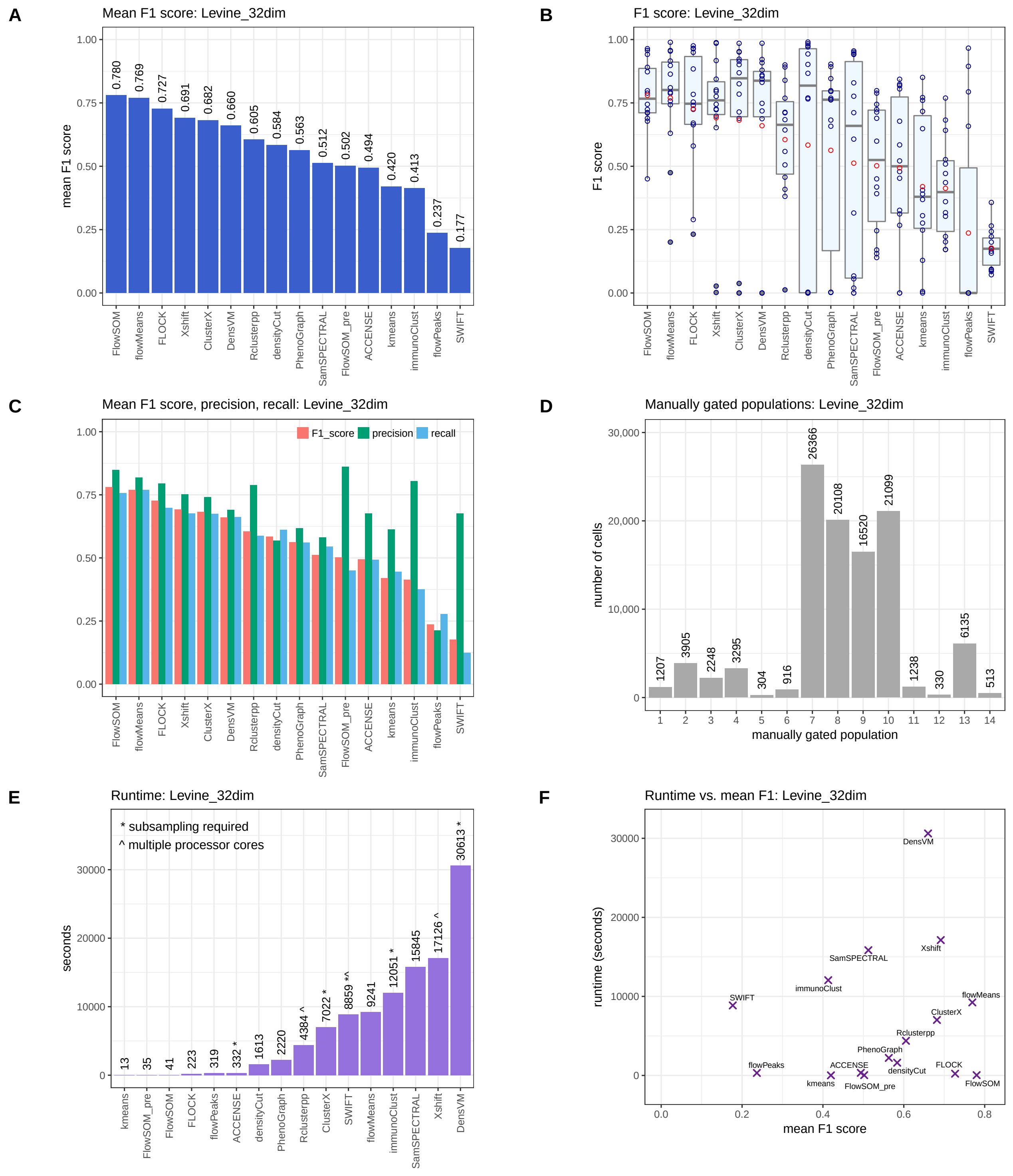


Figure 1. Results of comparison of clustering methods for data set Levine_32dim. (A) Mean F1 score across cell populations. (B) Distributions of F1 scores across cell populations. The box plots show medians, upper and lower quartiles, whiskers extending to 1.5 times the interquartile range, and outliers, with means shown additionally in red. (C) Mean F1 scores, mean precision, and mean recall. (D) Number of cells per reference population. (E) Runtimes. (F) Runtime vs. mean F1 score; methods combining high mean F1 scores with fast runtimes are seen toward the bottom-right.

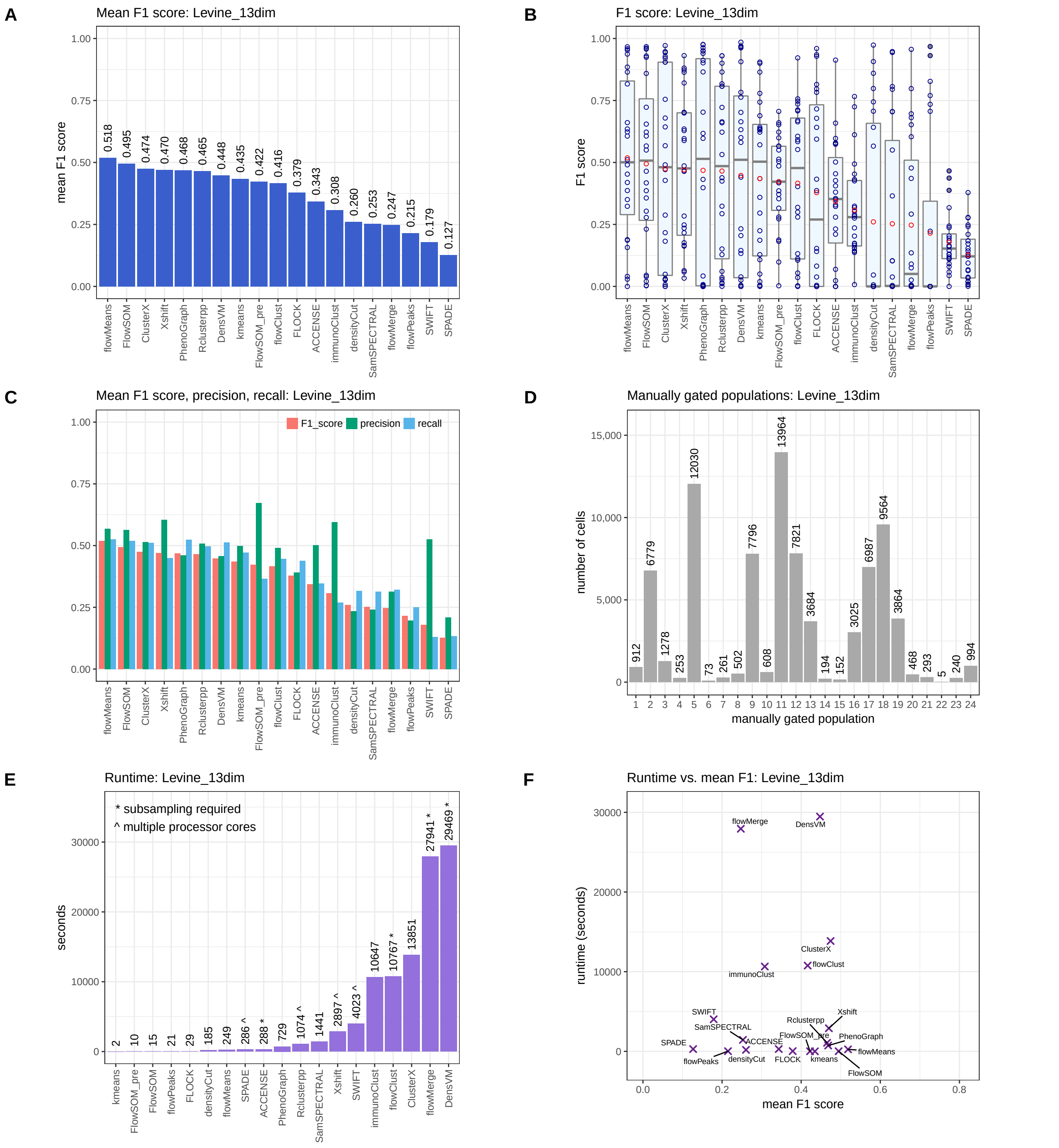


Figure 2. Results of comparison of clustering methods for data set Levine_13dim. (A) Mean F1 score across cell populations. (B) Distributions of F1 scores across cell populations. The box plots show medians, upper and lower quartiles, whiskers extending to 1.5 times the interquartile range, and outliers, with means shown additionally in red. (C) Mean F1 scores, mean precision, and mean recall. (D) Number of cells per reference population. (E) Runtimes. (F) Runtime vs. mean F1 score; methods combining high mean F1 scores with fast runtimes are seen toward the bottom-right.

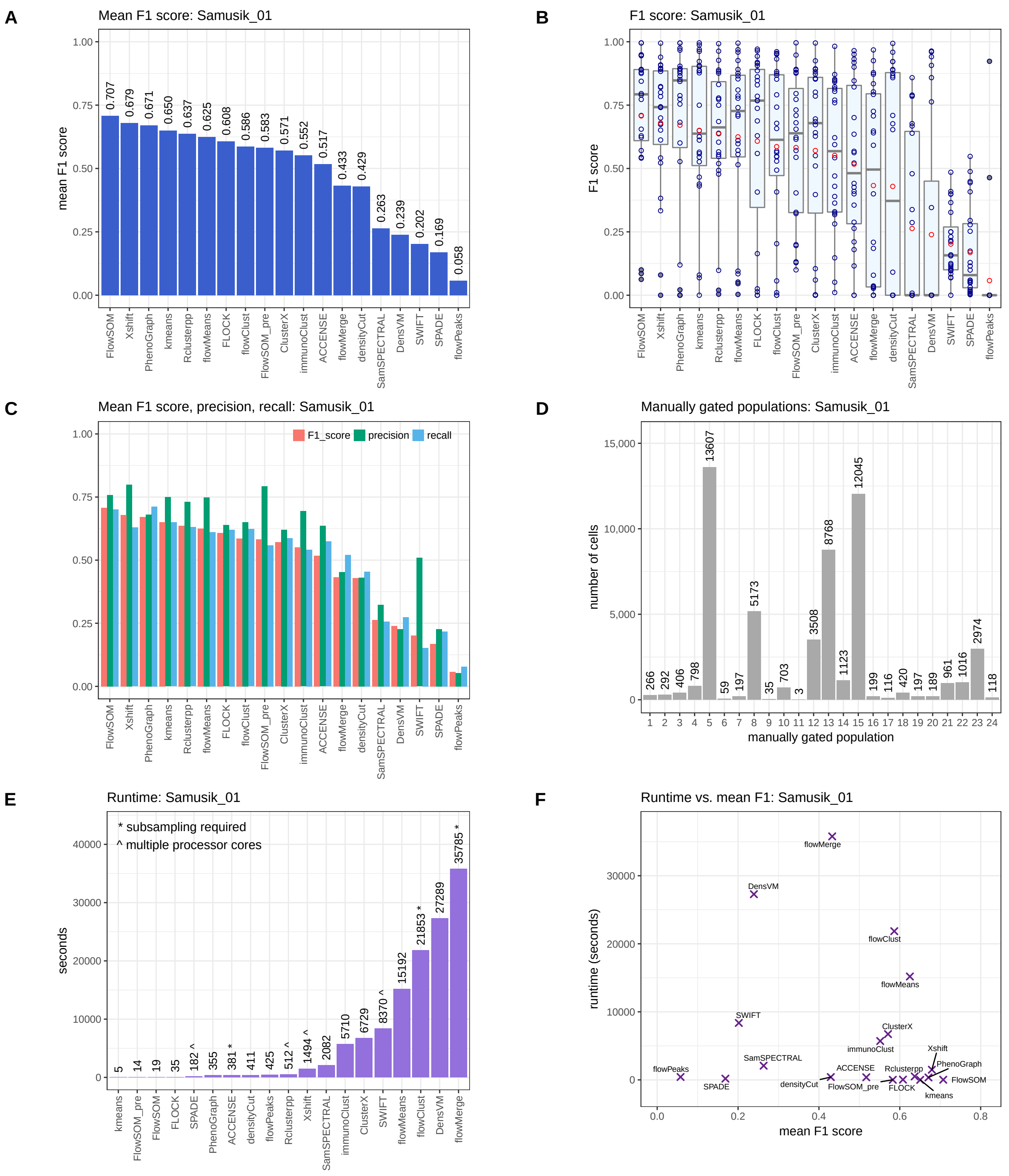


Figure 3. Results of comparison of clustering methods for data set Samusik_01. (A) Mean F1 score across cell populations. (B) Distributions of F1 scores across cell populations. The box plots show medians, upper and lower quartiles, whiskers extending to 1.5 times the interquartile range, and outliers, with means shown additionally in red. (C) Mean F1 scores, mean precision, and mean recall. (D) Number of cells per reference population. (E) Runtimes. (F) Runtime vs. mean F1 score; methods combining high mean F1 scores with fast runtimes are seen toward the bottom-right.

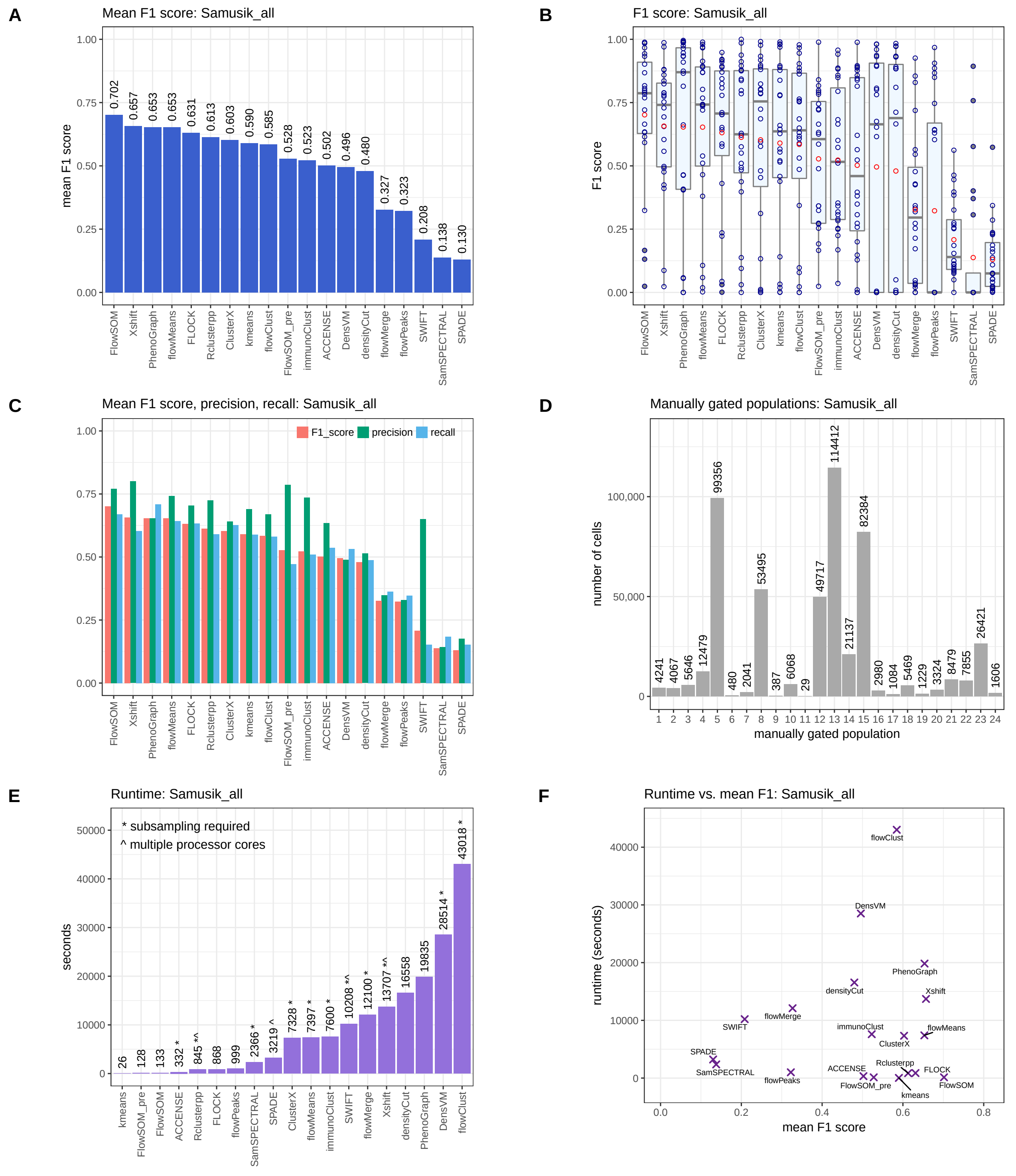


Figure 4. Results of comparison of clustering methods for data set Samusik_all. (A) Mean F1 score across cell populations. (B) Distributions of F1 scores across cell populations. The box plots show medians, upper and lower quartiles, whiskers extending to 1.5 times the interquartile range, and outliers, with means shown additionally in red. (C) Mean F1 scores, mean precision, and mean recall. (D) Number of cells per reference population. (E) Runtimes. (F) Runtime vs. mean F1 score; methods combining high mean F1 scores with fast runtimes are seen toward the bottom-right.

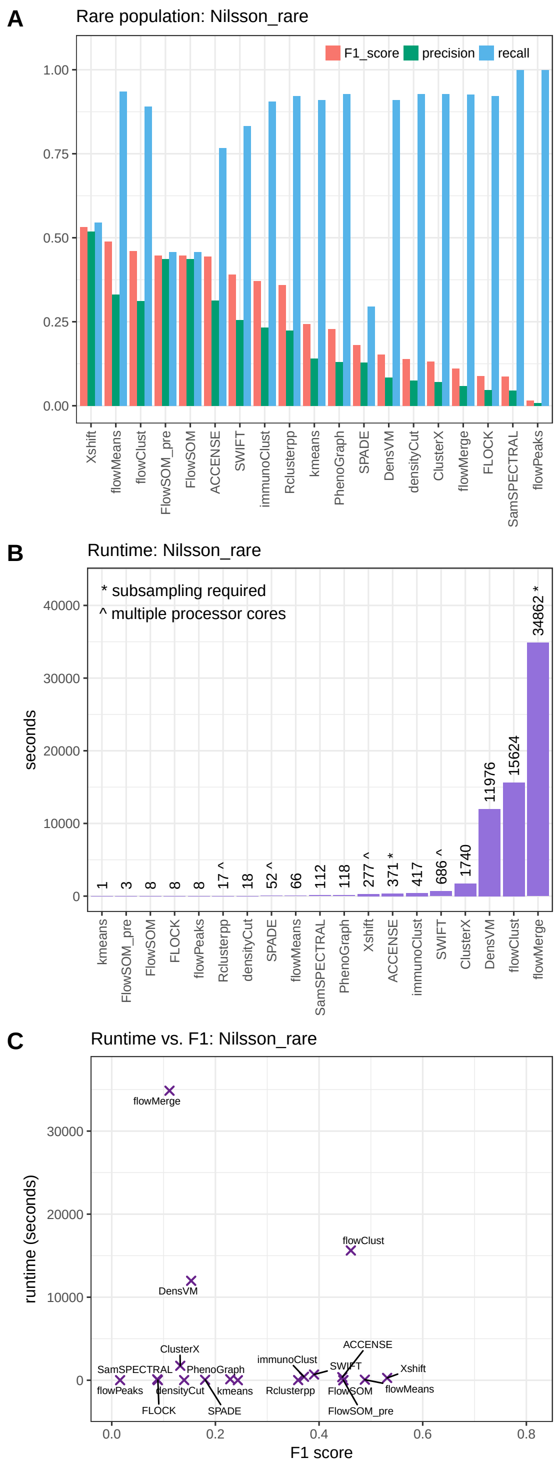


Figure 5. Results of comparison of clustering methods for data set Nilsson_rare. (A) F1 score, precision, and recall for the rare cell population of interest. The rare population contains approximately 0.8% of total cells. (B) Runtimes. (C) Runtime vs. F1 score; methods combining high F1 scores with fast runtimes are seen toward the bottom-right.

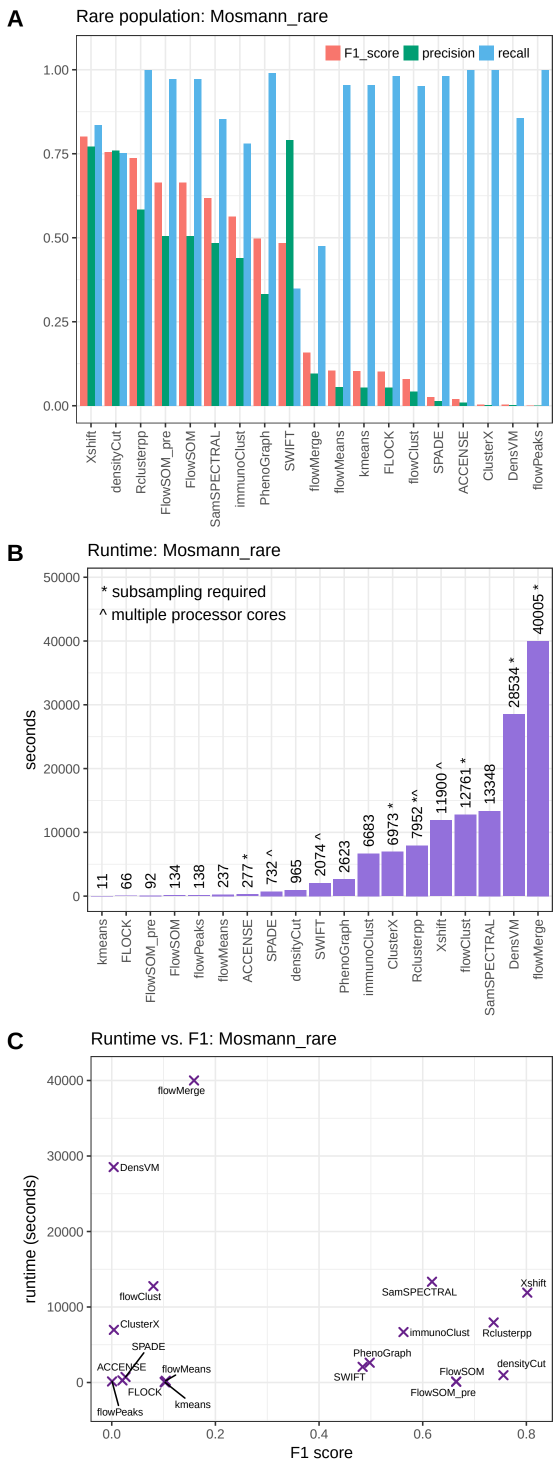
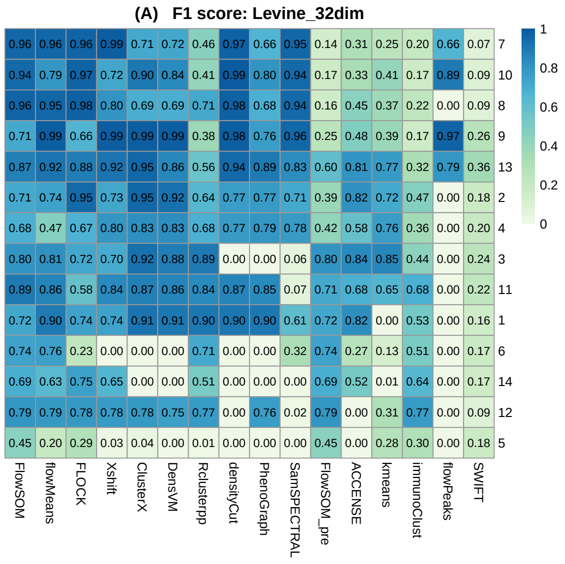
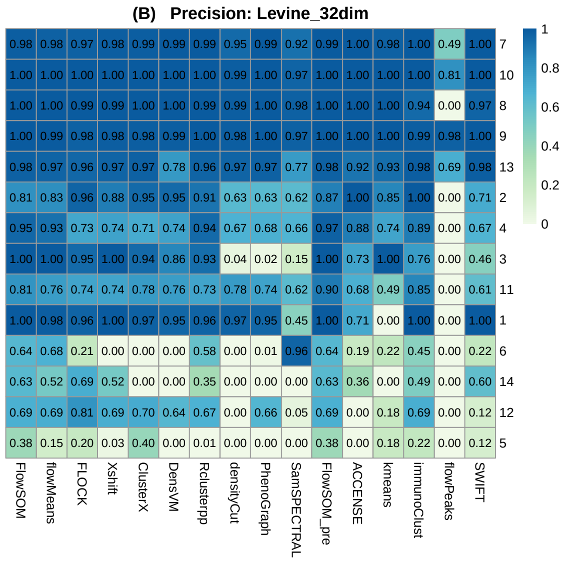


Figure 6. Results of comparison of clustering methods for data set Mosmann_rare. (A) F1 score, precision, and recall for the rare cell population of interest. The rare population contains approximately 0.03% of total cells. (B) Runtimes. (C) Runtime vs. F1 score; methods combining high F1 scores with fast runtimes are seen toward the bottom-right.

Figures: Additional results for individual populations





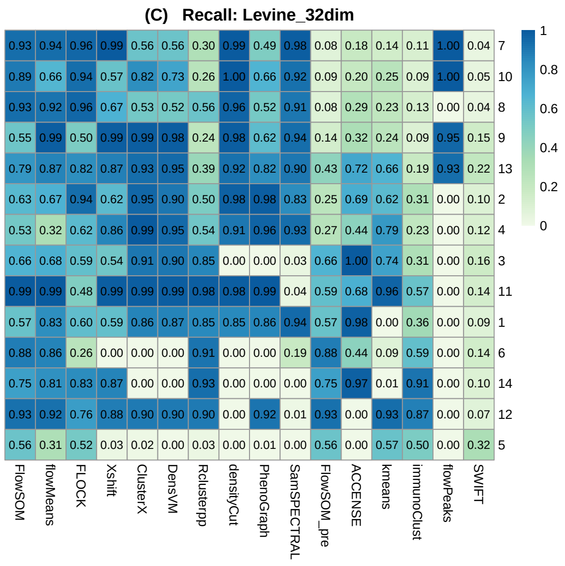
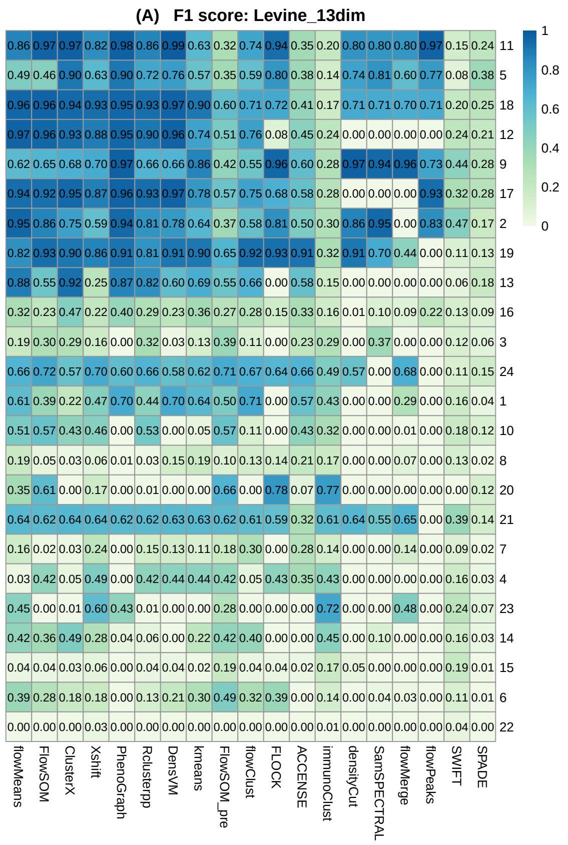
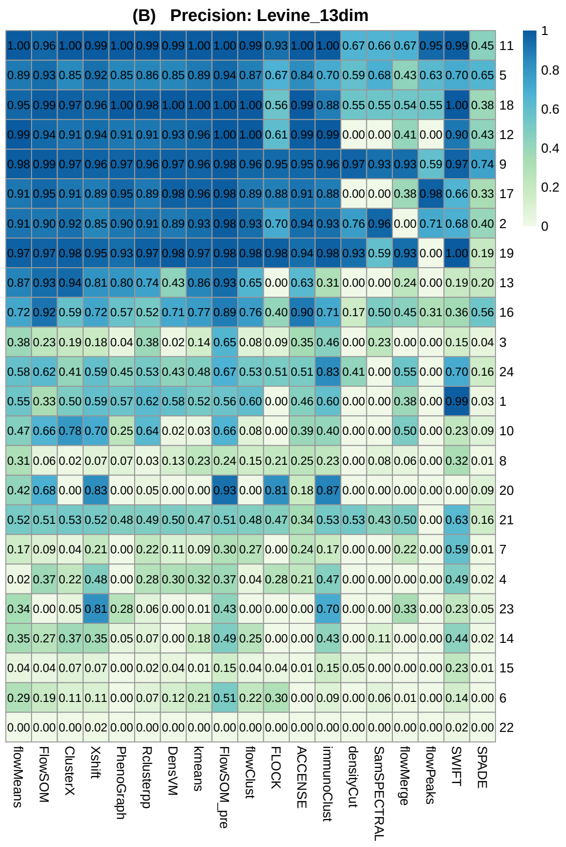


Figure 7. Additional results for individual cell populations (ordered by size) for data set Levine_32dim. Heatmaps show F1 scores (A), precision (B), and recall (C) for each reference population (manually gated population). Rows (populations) are ordered by decreasing population size, with the largest population at the top. Columns (clustering methods) are ordered by mean F1 score, as in the main results.





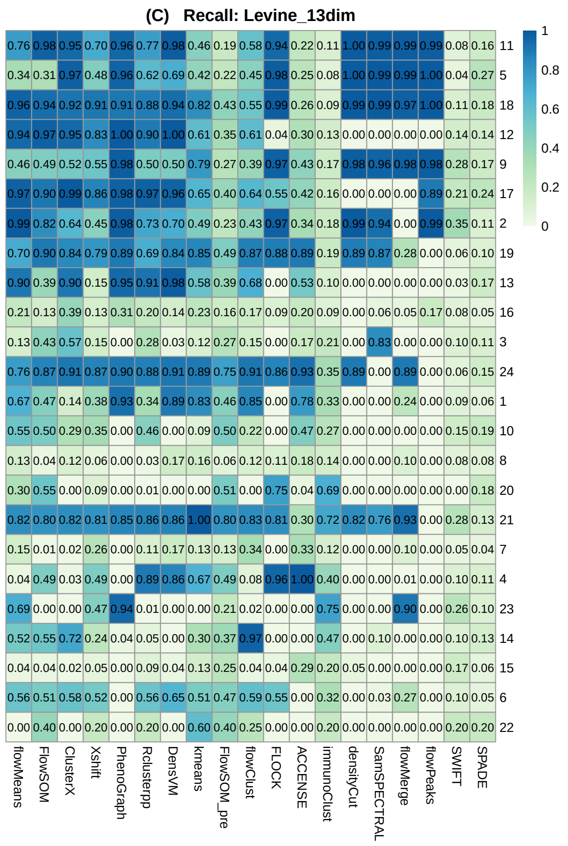
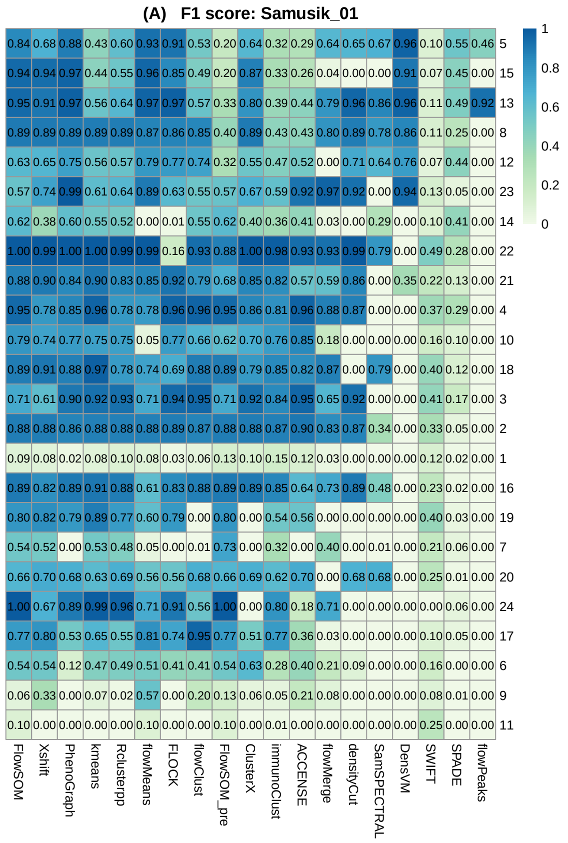
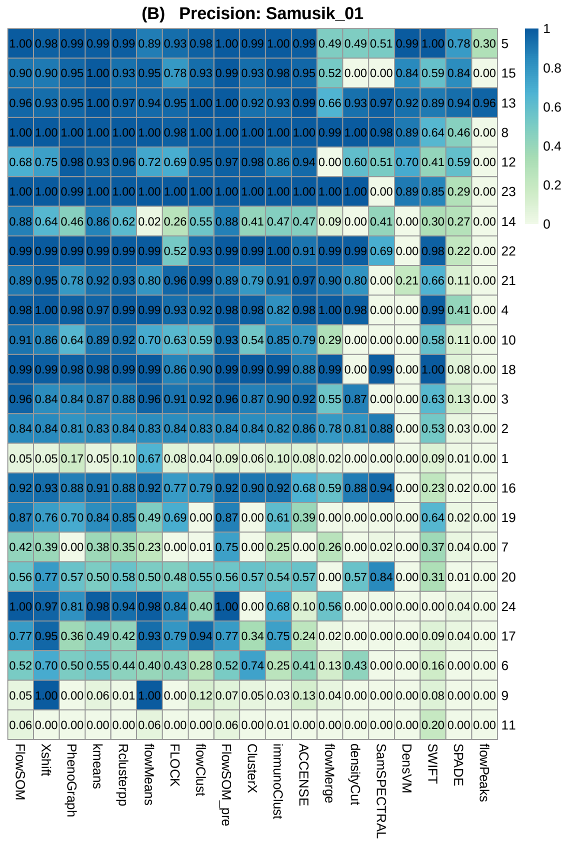


Figure 8. Additional results for individual cell populations (ordered by size) for data set Levine_13dim. Heatmaps show F1 scores (A), precision (B), and recall (C) for each reference population (manually gated population). Rows (populations) are ordered by decreasing population size, with the largest population at the top. Columns (clustering methods) are ordered by mean F1 score, as in the main results.





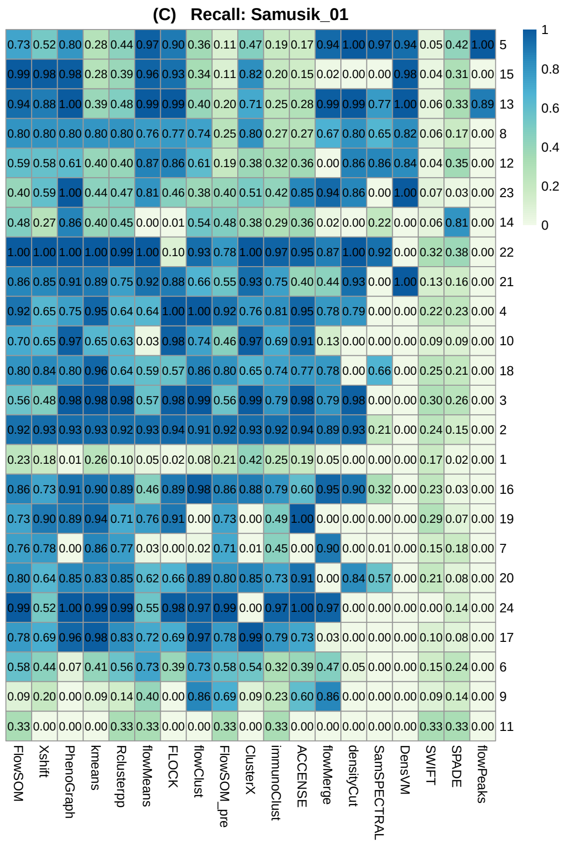
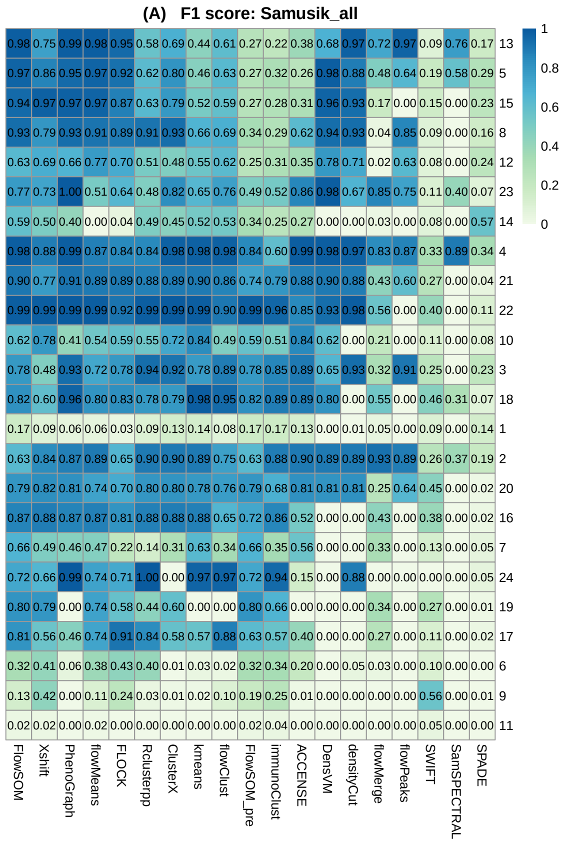
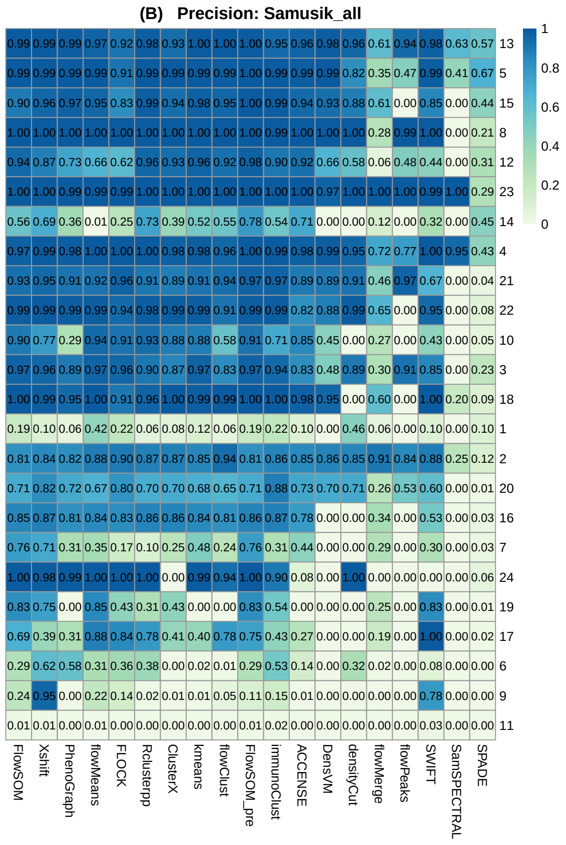


Figure 9. Additional results for individual cell populations (ordered by size) for data set Samusik_01. Heatmaps show F1 scores (A), precision (B), and recall (C) for each reference population (manually gated population). Rows (populations) are ordered by decreasing population size, with the largest population at the top. Columns (clustering methods) are ordered by mean F1 score, as in the main results.





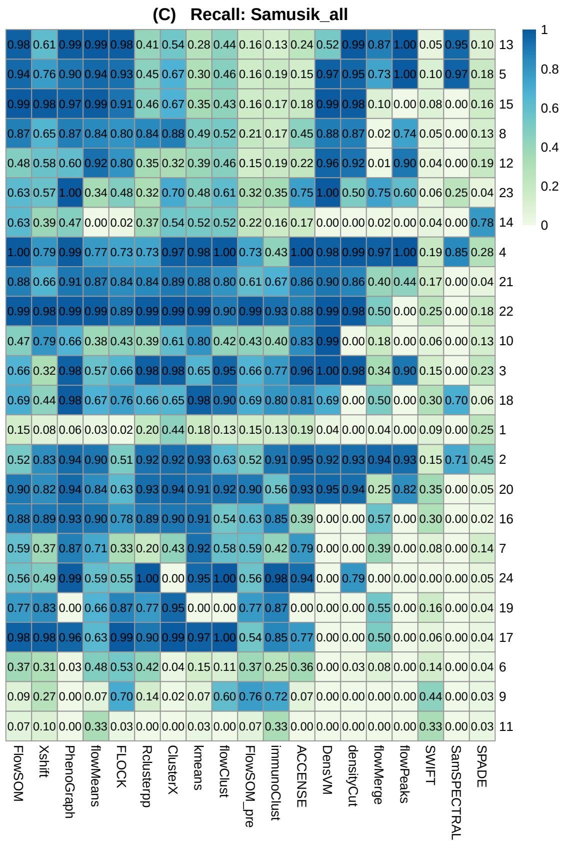


Figure 10. Additional results for individual cell populations (ordered by size) for data set Samusik_all. Heatmaps show F1 scores (A), precision (B), and recall (C) for each reference population (manually gated population). Rows (populations) are ordered by decreasing population size, with the largest population at the top. Columns (clustering methods) are ordered by mean F1 score, as in the main results.

Figures: Cluster median expression profiles

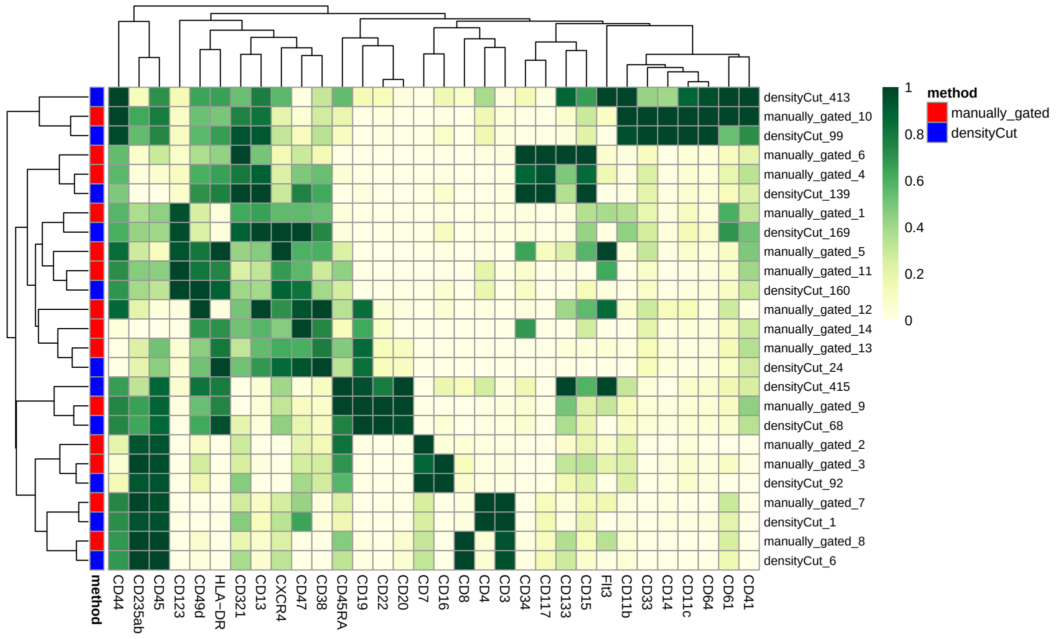


Figure 11. Median expression profiles of detected clusters and reference populations, densityCut, data set Levine_32dim. Heatmap shows median expression intensities of each protein marker (columns), for each detected cluster or reference population (rows). Values are arcsinh-transformed, and scaled between 0 and 1 for each protein marker. Rows and columns are sorted by hierarchical clustering (Euclidean distance, average linkage). Cluster and population indices are included in row headings. Red labels indicate rows representing reference populations, and blue labels indicate clusters detected by `densityCut`.

5 Discussion: Clustering method *densityCut*

Overall performance

Our evaluations showed that `densityCut` gave intermediate clustering performance overall for these data sets (Table 3). According to the mean F1 score (multiple cell populations) and F1 score (single rare cell population) criteria, `densityCut` ranked 8th, 14th, 14th, 14th, 14th, and 2nd for data sets `Levine_32dim`, `Levine_13dim`, `Samusik_01`, `Samusik_all`, `Nilsson_rare`, and `Mosmann_rare` respectively (out of 19 total methods for all data sets except `Levine_32dim`, and 16 total for `Levine_32dim`). In particular, very good performance was achieved for the single rare cell population of interest in data set `Mosman_rare` (ranked 2nd, with a mean F1 score of 0.756); but this level of performance was not repeated across the other data sets.

For the data sets with multiple populations of interest (`Levine_32dim`, `Levine_13dim`, `Samusik_01`, and `Samusik_all`), the lowest F1 scores for individual populations tended to occur for the relatively smaller populations (Figures 7–10), similar to the results observed for the other methods in the main paper.

Runtimes were intermediate (Table 3). Notably, runtimes were fast enough that all data sets could be analyzed without any subsampling, which is important for detecting rare cell populations (Table 2). Multiple processor cores were also not required (Table 2). The slowest runtimes were recorded for data set `Samusik_all`, which contained the largest number of cells (841,644 cells; approximately 4.5 hours).

For the two additional data sets from the FlowCAP-I challenges (Aghaeepour et al., 2013) (`FlowCAP_ND` and `FlowCAP_WNV`), results were similar to the other clustering methods (Table 4). See the main paper for a discussion of the differences between the two evaluation methodologies used for these data sets.

Number of clusters

The `densityCut` algorithm determines an optimal number of clusters automatically. As discussed above, two parameters (K and $alpha$) can be used to indirectly adjust the number of clusters. However, similar to many other methods, it is relatively difficult to precisely fine-tune the number of clusters by adjusting the indirect parameters (see Discussion in main paper). In addition, we note that, in biological terms, it is difficult to define a "true" number of cell populations in cytometry data, since cell populations may be viewed as forming a near-continuous progression of phenotypes (see Discussion in main paper).

We evaluated `densityCut` using the automatic number of clusters, based on default parameter settings, since this gave reasonable results overall (the same strategy as used in our main paper). However, for the data sets with multiple populations of interest, the automatic numbers were somewhat too small (Table 1; Table 2 in main paper). This limited the overall performance for these data sets, since F1 scores were equal to zero for the individual populations with no matching clusters.

Evaluation methodology

As mentioned above, the authors of `densityCut` (Ding et al., 2016) evaluated performance on CyTOF data using three data sets derived from the healthy bone marrow data in Levine et al. (2015), which is the same source we used to generate data sets `Levine_32dim` and `Levine_13dim`. However, the overall performance according to our evaluations for these data sets was not as good as that reported by Ding et al. (2016).

The differences are largely explained by differences in the evaluation methodologies. There were three main differences in our methodologies:

Clustering on all cells: In our evaluations, we ran clustering algorithms on all cells in the data sets (including "unassigned" cells, i.e. those not assigned to any population by manual gating), and evaluated performance on the cells where manually gated population labels were available (see main paper, Materials and Methods). This was also consistent with the methodology used by Levine et al. (2015). Manually gated population labels were available for 39% and 49% of cells in data sets `Levine_32dim` and `Levine_13dim` (Table 2 in main paper). By contrast, Ding et al. (2016) performed clustering on the assigned cells only. Clustering only assigned cells is problematic for two reasons: (i) it is likely to make the clustering task "easier" for the algorithm, since cells that are difficult to classify have been excluded from the data set; and (ii) it is difficult to interpret the reported performance when analyzing new experimental data sets, since manually gated population labels (and therefore a split into assigned and unassigned cells) will not be available for new data sets.

Weighting populations equally: For the data sets with multiple cell populations of interest, our evaluation methodology calculates mean F1 scores, with populations weighted equally (we calculate individual F1 scores for each reference population, and then take the unweighted average across the reference populations; see main paper, Materials and Methods). By contrast, the three evaluation metrics reported by Ding et al. (2016) — maximum-matching measure (MMM), normalized mutual information (NMI), and adjusted Rand index (ARI) — each give equal weighting to *cells* instead of clusters or populations (Ding et al., 2016, Supplementary Material). As argued in our main paper, we believe our methodology is more appropriate for cytometry data, since it ensures that both large and small populations are represented equally in the evaluations, and avoids the possibility that the overall scores are dominated by a small number of large clusters with high individual scores. (Both our methodology and that of Ding et al. (2016) also ensure that multiple mappings for each population or cell are not allowed; see main paper).

Combining individuals in data set Levine_32dim: Our data set `Levine_32dim` consists of bone marrow cells from two healthy donors, labeled H1 and H2. We performed clustering on the combined data from both individuals for this data set. By contrast, Ding et al. (2016) ran the clustering and evaluated performance separately for each individual. Therefore, our results are not precisely comparable for this data set.

References

Aghaeepour, N., Finak, G., Hoos, H., Mosmann, T. R., Brinkman, R., Gottardo, R., and Scheuermann, R. H. (2013). Critical assessment of automated flow cytometry data analysis techniques. *Nature Methods*, 10(3):228–238.

Ding, J., Shah, S., and Condon, A. (2016). densityCut: an efficient and versatile topological approach for automatic clustering of biological data. *Bioinformatics*, 32(17):2567–2576.

Levine, J. H., Simonds, E. F., Bendall, S. C., Davis, K. L., Amir, E.-a. D., Tadmor, M. D., Litvin, O., Fienberg, H. G., Jager, A., Zunder, E. R., Finck, R., Gedman, A. L., Radtke, I., Downing, J. R., Pe'er, D., and Nolan, G. P. (2015). Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell*, 162:184–197.

Weber, L. M. and Robinson, M. D. (2016). Comparison of Clustering Methods for High-Dimensional Single-Cell Flow and Mass Cytometry Data. *Cytometry Part A*, 89A:1084–1096.