Comparison of Clustering Methods for Multiparametric Cytometry Data Analysis in order to Implement an R/Shiny Application

Anna Guadall Roldán
Máster universitario en Bioinformática y Bioestadística (UOC, UB)
Desarrollo de herramientas de soporte a la ómica

Codirectores externos (Institut de Recherche Saint-Louis, Paris):
Simon Tournier
Sophie Duchez

Consultor:
Antonio Jesús Adsuar Gómez

Profesor responsable:
Javier Luis Cánovas Izquierdo

Junio de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Comparison of Clustering Methods for Multiparametric Cytometry Data Analysis in order to Implement an R/Shiny Application*
Nombre del autor:	*Anna Guadall Roldán*
Nombre de los codirectores externos:	*Simon Tournier* *Sophie Duchez*
Nombre del consultor/a:	*Antonio Jesús Adsuar Gómez*
Nombre del PRA:	*Javier Luis Cánovas Izquierdo*
Fecha de entrega (mm/aaaa):	06/2019
Titulación:	Máster universitario en Bioinformática y Bioestadística (UOC, UB)
Área del Trabajo Final:	*Desarrollo de herramientas de soporte a la ómica*
Idioma del trabajo:	*Inglés*
Palabras clave	*Multiparametric Cytometry* *Unsupervised Clustering* *Algorithm Performance*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La citometría de flujo convencional es una tecnología que permite detectar hasta 30 parámetros por célula. Recientemente, la citometría de flujo y la espectrometría de masas se han fusionado dando lugar a la denominada citometría de masas, que potencialmente permite la detección de hasta 100 parámetros por célula. Las poblaciones celulares se caracterizan principalmente mediante el procedimiento de *gating*, consistente en delimitar manualmente las poblaciones usando histogramas o gráficos de puntos de manera secuencial. Este procedimiento es lento, impreciso y particularmente inadecuado para un elevado número de parámetros. En los últimos años se han estado desarrollando nuevas técnicas computacionales con la finalidad de manejar datos de citometría multidimensional de modo eficiente. Sin embargo, la eficacia de tales desarrollos todavía se está evaluando. Además, el manejo de estas técnicas requiere habilidad en el uso de paquetes R y programación.

El objetivo principal de este proyecto es proporcionar a los citometristas algoritmos de aprendizaje no supervisado y técnicas de visualización para explorar datos de citometría multiparamétrica de modo reproducible. Con esta finalidad, se ha realizado una extensa búsqueda bibliográfica sobre algoritmos de agrupamiento aplicados a la citometría y se ha desarrollado una metodología para la evaluación del rendimiento. Una selección de algoritmos ha sido contrastada aplicando esta metodología a datos de citometría reales y datos ficticios generados expresamente con este fin. Este estudio comparativo ha permitido seleccionar un algoritmo de agrupamiento, RPhenograph, para ser implementado mediante una aplicación Shiny. La metodología desarrollada es aplicable para la evaluación de nuevos algoritmos y nuevos diseños experimentales.

Abstract (in English, 250 words or less):

Conventional flow cytometry is an experimental technique enabling to measure up to 30 fluorescence parameters per cell. Recently, flow cytometry has been fused to mass spectrometry giving rise to a new methodology named mass cytometry that can potentially detect up to 100 parameters per cell. Cell populations are mainly characterized by a procedure known as gating, consisting in manually delimitating cell subsets using histograms or two-dimensional dot plots in a sequential manner. This procedure is time-consuming, imprecise and particularly inadequate to be used with a high number of parameters. In the past few years new computational techniques have been developed in order to efficiently handle high-dimensional cytometry data. However, such developments are still under evaluation. Furthermore, dealing with these techniques requires proficiency in using R packages and script writing.

The main objective of this project is to provide cytometrists with efficient and easy-to-use unsupervised learning algorithms and visualization tools to explore high-dimensional cytometry data in a reproducible way. To that end, an extensive bibliographic research on unsupervised clustering algorithms applied to cytometry data has been performed and a methodology for performance evaluation has been developed. A selection of algorithms has been benchmarked using this methodology and both real cytometry and synthetic data, the latter being specially generated to that end. This comparative study has allowed the selection of a clustering algorithm, RPhenograph, to implement a Shiny application. The developed methodology is now ready to be applied to benchmark further algorithms and compare performances on other experimental designs.

Contents

1 Introduction **4**
1.1 Background 4
1.2 Objectives 5
1.3 Scope and Methods 6
1.4 Implementation and timetable 8
1.5 Obtained products 11
1.6 Project structure 12

2 First selection of unsupervised clustering algorithms based on bibliographic review **13**
2.1 Bibliographic research 13
2.2 Selection of algorithms 18

3 Methods for performance evaluation **20**
3.1 F1 score 20
3.2 CPU time 28

4 Second selection of unsupervised clustering algorithms based on testing on synthetic data **29**
4.1 Clustering algorithms 29
4.2 R script to generate synthetic data 29
4.3 Synthetic data generated for algorithm benchmarking 33
4.4 Algorithm benchmarking script 34
4.5 Evaluation of performance 37
4.6 Conclusions 42

5 Comparative study of selected clustering algorithms **44**
5.1 5-parameter design 44
5.2 11-parameter design 62
5.3 Conclusions 68

6 Discussion **71**

7 Conclusions **72**

8 Glossary **73**

9 References **74**

List of Figures

1 Gantt chart with the project schedule. 10

2 t-SNE mapping of the original five-dimensional data. Data points are colored according reference cell labels (upper left), the RPhenograph clustering (upper right), and the partitions resulting from the two matching procedures (bottom). 27

3 F1 scores obtained for sample "3 A" for all the cell types using RPhenograph, flowMeans or flowPeaks. Dot sizes and coloring both indicate populations' frequencies. Missing points on a row indicate undetected populations. 37

4 Mean F1 scores for all the samples generated using RPhenograph, flowMeans or flowPeaks. 38

5 Mean F1 scores for all the samples (4 conditions performed in duplicate) related to the number of predicted clusters obtained using RPhenograph, flowMeans or flowPeaks. Dotted lines indicate the number of populations (ground truth). 39

6 F1 scores obtained for sample "3 A" for all the cell types using flowSOM and the dimensionality reduction algorithms t-SNE and UMAP followed by flowMeans or flowPeaks clustering methods. Coloring indicates the populations' frequencies. n = 5 random starts for every method. 40

7 Mean F1 scores for all the samples generated using flowSOM and the dimensionality reduction algorithms t-SNE and UMAP followed by flowMeans, flowPeaks or ClusterX clustering methods. n = 5 random starts for every method. 41

8 Mean F1 scores for all the samples (4 conditions performed in duplicate) related to the number of predicted clusters obtained using flowSOM and the dimensionality reduction algorithms t-SNE and UMAP followed by flowMeans, flowPeaks or ClusterX clustering methods. n = 5 random starts for every method. Dotted lines indicate the number of populations (ground truth). . . 43

9 Manual gating strategy for the 5-parameter experimental design. The five main populations are indicated in orange. 45

10 Representation of the hierarchical gating strategy for the 5-parameter design imported from FlowJo. 46

11 Cell frequencies for the 3 samples analyzed for the 5-parameter design, obtained by manual gating. Frequencies for the 5 main populations and the potential 20 subpopulations are indicated, as well as the outliers. 54

12 5-parameter design. FlowSOM clustering has been performed specifying to find 4, 5, 8, 10 or 20 clusters. Sample 1, n=2 down-samples; samples 2 and 3, n=3 down-samples. 56

13 5-parameter design. Mean F1 scores related to the number of predicted clusters (left panel) and the number of cells (right panel, mean ± standard error). Sample1, n=2; samples 2 and 3, n=3. FlowSOM, RPhenograph and UMAP + flowPeaks, 25,000 and 50,000 cells, n=3; 100,000 cells, n=2. UMAP + ClusterX 25,000 and 50,000 cells, n=2; 100,000 cells, n=1. 57

14 5-parameters design. Number of matched partitions (detected populations) related to the number of clusters. Sample 1, n=2; samples 2 and 3, n=3. FlowSOM, RPhenograph and UMAP + flowPeaks, 25,000 and 50,000 cells, n=3; 100,000 cells, n=2. UMAP + ClusterX 25,000 and 50,000 cells, n=2; 100,000 cells, n=1. 58
15 5-paramenter design. CPU (user) runtimes related to the number of detected clusters (left panel) and the number of cells (right panel). Sample 1, n=2; samples 2 and 3, n=3. FlowSOM, RPhenograph and UMAP + flowPeaks, 25,000 and 50,000 cells, n=3; 100,000 cells, n=2. UMAP + ClusterX 25,000 and 50,000 cells, n=2; 100,000 cells, n=1. 59
16 5-parameter design. Anlysis of subpopulations on split main populations. 1st and 2nd panels: 25,000 cell down-samples. 3th and 4th panels: 50,000 cell down-samples. 1st and 3th panels: The F1 scores are indicated for all the matched partitions. 2nd and 4th panels: The F1 scores obtained for the matched partitions corresponding to the main populations are related to the mean F1 scores computed for the populations that have been split in more than one cluster. Dot, square and triangle sizes are drown proportional to the number of clusters found for each population. Notice that NO stands for NO_BTNK cells. 61
17 Manual gating strategy for the 11-parameter experimental design. 19 populations of lymphocytes are delimited (orange). 63
18 Representation of the hierarchical gating strategy for the 11-parameter design imported from FlowJo. 64
19 Cell frequencies for the 2 samples analyzed for the 11-parameter design, obtained by manual gating. Frequencies for 19 populations are indicated, as well as the outliers. 65
20 11-parameter design. FlowSOM clustering has been performed specifying to find 5, 10, 15, 20, 30, 40, 50 or 60 clusters. 66
21 11-parameter design. Mean F1 scores related to the number of predicted clusters (left panel) and the number of cells (right panel, mean ± standard error). Samples 1 and 2, n=4 down-samples. 67
22 11-parameters design. Number of matched partitions (detected populations) related to the number of clusters. Samples 1 and 2, n=4 down-samples. . . . 68
23 11-paramenter design. CPU (user) runtimes related to the number of detected clusters (left panel) and the number of cells (right panel).Samples 1 and 2, n=4 down-samples. 69
24 Summary of the results achieved with the methods and conditions tested. 5_2: 5 parameters, sample 2. 5_3: 5 parameters, sample 3. 11_1: 11 parameters, sample 1. 11_2: 11 parameters, sample 2. 70

List of Tables

1 Summary of Accomplishment: **Milestones**
(Milestones scheduled for Phase II are indicated in bold). 9
2 R/Bioconductor packages for cytometry data processing 13
3 Dimensionality reduction, unsupervised clustering and/or visualization methods 15
4 Unsupervised clustering methods 16
5 Other approaches and purposes 17
6 Recent reviews and benchmarking articles on high-dimensional cytometry data analysis 18
7 First selected clustering algorithms 29
8 Phenotypes 33
9 Parameters used for synthetic sample generation 33
10 Cell populations percentages 34

1 Introduction

1.1 Background

1.1.1 General description

This project aims to compare new standardized protocols based on emerging computing strategies for multiparametric cytometry data analysis.

1.1.2 Project justification

The *Institut de Recherche Saint-Louis (IRSL)* in Paris is a reference European research institute focused on hematology, oncology and immunology. The IRSL has developed state-of-the-art core facilities in order to provide scientists with highly specialized research equipment. The facility, managed by Dr Setterblad, is organized in four main axes: Flow Cytometry, Imaging, Genomics and the BioData Center. The facility's team offers technical support and scientific advice but, unless for some specific cases, does not perform nor analyze user's experiments. In order to help users to work more independently, seminars discussing technical aspects in data acquisition and analysis methods are often organized.

The use of the Flow Cytometry facility has dramatically increased in the last few years. Flow cytometry is an experimental technique that enables the measurement of multiple parameters at a single cell level with high accuracy and at a high speed (thousands of cells per second). Cells can be fluorescently stained, typically with fluorochromes conjugated to antibodies directed against cell surface or intracellular molecules. Individual cell fluorescence, as well as information about cell size and structural complexity, is collected thanks to a system of lasers and photonic detectors. In hematology, flow cytometry is routinely used to characterize and quantify cell populations. This technology, developed in the late sixties/early seventies, has evolved during the past decades mainly by increasing the number of lasers, enabling to simultaneously measure up to 30 fluorescence parameters per cell. Additionally, flow cytometry has been recently fused to mass spectrometry giving rise to a new technology, named mass cytometry (or *cytometry by time-of-flight*, CyTOF), that can potentially detect up to 100 parameters per cell using metal-conjugated antibodies. The possibility to acquire such a number of parameters has opened up new research and diagnostic perspectives, particularly in the quest for rare populations [1].

Users of the IRSL Flow Cytometry facility have access to flow cytometers that can detect up to 18 fluorescent signals. The facility also offers informatic equipment to analyze the data generated in the same facility or in other facilities that enable to measure even a higher number of parameters.

To date, cell populations are characterized by a procedure known as *gating*, consisting in manually delimitating cell subsets using histograms or two-dimensional dot plots in a sequential manner. `FlowJo` is the reference software used to perform these tasks. Even though it has a friendly graphical environment, `FlowJo` offers such a number of functions that even experienced cytometrists are encouraged to attend a specific training to properly use it.

Gating can be an appropriate procedure to handle a low number of parameters and/or experimental conditions; however, it is not efficient to handle high-dimensional datasets [2]. This procedure is time-consuming and particularly affected by inter-user variability. Moreover, the choice of the parameters to be used to define a cell population, and the specific sequential order by which these parameters are used in the gating strategy, can alter the final result. This can be particularly troublesome when looking for rare populations. **There is therefore a need for new computational techniques that can handle high-dimensional cytometry data in a more efficient and reproducible way.** In the past few years new promising developments have appeared in the literature. New analysis strategies include different techniques for automatic gating, clustering, dimensionality reduction and data visualization [1]. However, such developments are still under evaluation and **there is no standardized protocol approved by the cytometric community**.

Dr Duchez, responsible of the Flow Cytometry facility, and Dr Tournier, responsible of the BioData Center, are gathering efforts to test these emerging analysis tools and to introduce them to the Flow Cytometry facility's users. However, **dealing with these techniques requires proficiency in using R packages and script writing, a hard task for experimental cytometrists that feel uncomfortable with the command line**.

1.2 Objectives

The main objective of this project is to **provide cytometrists with efficient and easy-to-use unsupervised learning algorithms and visualization tools to explore high-dimensional cytometry data in a reproducible way**. Further applications that could also be interesting for multiparametric cytometry analysis such as automated population identification or biomarker discovery are beyond the scope of this project.

1.2.1 Main objectives

1. Explore and select appropriate algorithms of unsupervised learning and data visualization for multiparametric cytometry data (8-18 parameters).
2. Make the selected computational approaches accessible to the Flow Cytometry facility's users with little or no knowledge in bioinformatics.

1.2.2 Specific objectives

Objective 1

1.1. Establish a standardized method to import pre-processed FCS (Flow Cytometry Standard) data files.

1.2. Establish some standardized methods to apply unsupervised learning techniques involving

clustering and dimensionality reduction methods.
1.3. Establish some standardized methods for data visualization.
1.4. Establish a standardized method to export the compensated, transformed, dimensionally-reduced and/or clustered data as FCS files.

Objective 2
2.1. Construct pipelines implementing the established methods that could be launched on batch and that could be used to build a graphical front-end.

1.3 Scope and Methods

1.3.1 Statements

The methods used to achieve the objectives of this project have been considered taking into account the following aspects:

1.3.1.1 Target

The final aim of this project, the graphical front-end, is intended to be used by the cytometrists of the IRSL Flow Cytometry facility (and by any other who may want to). IRSL users mostly include researchers interested in the detection, quantification and discovery of cell populations (including rare populations), with different experience levels in flow cytometry, an increasing interest in multiparametric cytometry, but almost no, if not any, programming skills.

1.3.1.2 Computing resources

The IRSL Cytometry facility offers its users three computers for post-acquisition analysis equipped with proprietary software such as `FlowJo` available upon reservation. Besides, the IRSL BioData Center has a Dell T620 machine with 32 processor cores and 32GB of RAM, and a cluster with 9 nodes, 40 processor cores per node and 120 GB of RAM per node.

1.3.1.3 Reproducibility and Portability

The pipelines developed on this project must be reproducible over time and portable over different computer environments. Users should be able to reanalyze data at any time with no need to adapt the code to different programming environments or software versions.

1.3.1.4 Collaborative work

This project is codirected by Dr Tournier and Dr Duchez from the IRSL Core Facilities, and supervised by Dr Adsuar from the UOC. There is a need for communicative tools that also enable to trace the development of the project.

1.3.1.5 Availability of code

Even though this project is designed to fulfill the needs of the IRSL Flow Cytometry facility users, it is freely available to use and open for feedback.

1.3.1.6 Education

This project aims not only to provide cytometrists an easy-to-use tool, but also to introduce them to programming.

1.3.2 Resolutions

Taken into consideration these circumstances and purposes, the following strategies have been chosen:

1.3.2.1 Choice of software

The software should be open source in order to allow IRLS users to use it in their own computers. In that way, the IRSL analysis computers would be exclusively used for the proprietary software. The language software should also enable to construct pipelines that could be launched on batch on the IRSL machines. Finally, the software should support the majority of algorithms designed for cytometry data analysis. For all these reasons, the `R` programing language and `RStudio` software, altogether with `Bioconductor` and other freely available packages, seem to be the perfect choice. Moreover, `RStudio` has a simple and efficient package to build interactive front-ends, `Shiny`, that enables users to easily use `RStudio` through a graphical interface while displaying the code behind. The latter is an important point, as one of the missions of the IRSL Core Facilities is to introduce researchers to bioinformatics.

1.3.2.2 Programming environment

`R` scripts have been developed in `R Markdown` through `RStudio`. All the work has been performed on a laptop (MacBook Pro with an Intel Core i7 processor and 16 GB of RAM). The final products of this project (`R` pipelines and, afterwards, a `Shiny` front-end) are intended to be introduced on `Docker` images containing all the necessary dependencies to run them (this aspect will not be part of the project).

1.3.2.3 Communication

Meetings are held once a week at the IRSL Core Facilities. Brief summaries of the meetings are shared on a free facility for project management (www.teamgantt.com), altogether with a Gantt chart of the project. In order to share, comment and keep a record of the script development and modifications, the code is saved under the distributed version control system `git` and served by a public platform at: https://github.com/plateforme-stlouis/RCyto.

1.3.2.4 Algorithm benchmarking

The algorithm selection has been performed through three main steps:

1. Algorithm selection based on the literature
2. Testing selected algorithms for biological significance on generated synthetic data
3. Evaluation of computational complexity (CPU time)

The use of synthetic data with has enabled testing the algorithms on controlled situations, making special attention to the detection of rare populations (<1%). Synthetic samples are based in real marker expression and simulate being spill-over compensated and pre-processed (i.e., cleaned from debris, dead cells and doublets).

1.3.2.5 Algorithm testing on real data

The selected algorithms have been tested on real multiparametric cytometry data obtained from IRSL experiments and pre-processed using `FlowJo`.

1.4 Implementation and timetable

1.4.1 Activities

Here below are listed the main activities planned for this project.

1. Bibliographic research on `R` packages for cytometry data processing.
2. Bibliographic research on algorithms for unsupervised learning and visualization methods for multiparametric cytometry data.
3. Bibliographic research on multiparametric cytometry data.
4. Selection of algorithms based on the literature.
5. Development of an `R` script that generates synthetic cytometry data.
6. Generation of synthetic cytometry data.
7. `R` script writing: Testing of selected algorithms and implementation on 'R pipelines.
8. Selection of algorithms based on their performances on synthetic data.
9. Testing of selected algorithms with real data.
10. Selection of algorithms based on their algorithm performance (CPU time).
11. Final selection, correction and/or validation of the `R` pipeline(s)
12. Development of a `Shiny` front-end that implements the validated `R` pipeline(s).

This list has been modified with respect to the initial Project Plan Proposal. In fact, it was first planned to make a third algorithm selection based on the CPU runtimes recorded with the synthetic data sets, just after activity 8. Arrived at that time point, it appeared inadequate to eliminate more algorithms while it was not known how would they perform with real data. Thus, the algorithms have been first tested with real data before performing any other selection. Both performance scores and time complexity have been taken into consideration for the final selection.

1.4.2 Timeline

The different activities and milestones had been scheduled ambitiously. Tasks and milestones were first delineated as indicated in Figure 1.

This scheduled has been modified during the project development. Indeed, the modification on the planned activities mentioned above impacted on the milestone plan, which was modified in the Phase I Project Implementation Report (see Table 1). Nevertheless, this schedule has not been accomplished. The dates of achievement that have been sensibly delayed or postponed are indicated in orange.

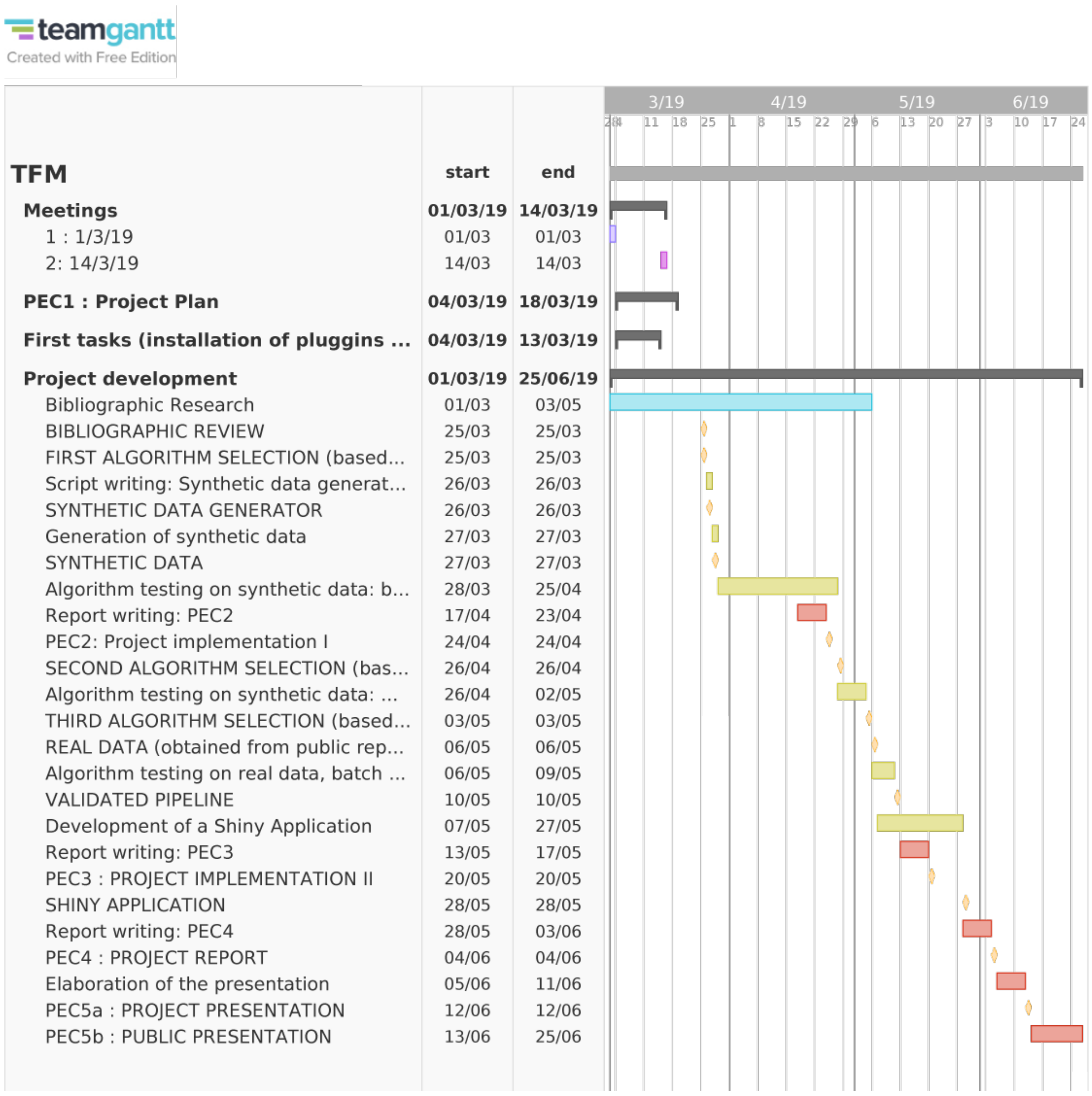
Table 1: Summary of Accomplishment: **Milestones**
(Milestones scheduled for Phase II are indicated in bold).

	Milestone	Scheduled	Achieved
1	Bibliographic review on algorithms for unsupervised learning and visualization methods for multiparametric cytometry analysis.	25/03/19	**27/03/19**
2	First algorithm selection (based on the bibliographic review).	25/03/19	**27/03/19**
3	R script to generate synthetic cytometry data.	26/03/19	**27/03/19**
4	Synthetic data.	27/03/19	**27/03/19**
5	**Second algorithm selection (based on the tests performed on synthetic data).**	**26/04/19**	14/05/19
6	**Testing on real data.**	**10/05/19**	27/05/19
7	**Third algorithm selection (based on performances on real data)**	**17/05/19**	29/05/19
8	**Shiny front-end.**	**28/05/19**	Cancelled

1.4.2.1 Justification of delay

The task that has generated the main delay has been establishing an adequate method to evaluate algorithm performances. A first procedure was used with the algorithms tested on synthetic data and refined when the tests with real data began. In fact, the algorithm selection based on synthetic data (fifth milestone) was finished ahead of time (on 19/04/2019) and was performed again with the improved evaluation method, which is discussed in Chapter 3. **Indeed, choosing adequate algorithms is the core of the project and it could not be overlooked.**

Delaying the establishment of the evaluation method and the test on synthetic data had consequences on the rest of the project, leading to postpone of the `Shiny` front-end. Although implementing a `Shiny` front-end would be quite useful for non-expert usability, it represented a less challenging work since the framework allows to easily glue straightforward code. Actually, **the main products of this project have been the evaluation and benchmarking methods, the obtained performance measures and the final conclusions.**



TFM	start	end
Meetings	**01/03/19**	**14/03/19**
1 : 1/3/19	01/03	01/03
2: 14/3/19	14/03	14/03
PEC1 : Project Plan	**04/03/19**	**18/03/19**
First tasks (installation of pluggins ...	**04/03/19**	**13/03/19**
Project development	**01/03/19**	**25/06/19**
Bibliographic Research	01/03	03/05
BIBLIOGRAPHIC REVIEW	25/03	25/03
FIRST ALGORITHM SELECTION (based...	25/03	25/03
Script writing: Synthetic data generat...	26/03	26/03
SYNTHETIC DATA GENERATOR	26/03	26/03
Generation of synthetic data	27/03	27/03
SYNTHETIC DATA	27/03	27/03
Algorithm testing on synthetic data: b...	28/03	25/04
Report writing: PEC2	17/04	23/04
PEC2: Project implementation I	24/04	24/04
SECOND ALGORITHM SELECTION (bas...	26/04	26/04
Algorithm testing on synthetic data: ...	26/04	02/05
THIRD ALGORITHM SELECTION (based...	03/05	03/05
REAL DATA (obtained from public rep...	06/05	06/05
Algorithm testing on real data, batch ...	06/05	09/05
VALIDATED PIPELINE	10/05	10/05
Development of a Shiny Application	07/05	27/05
Report writing: PEC3	13/05	17/05
PEC3 : PROJECT IMPLEMENTATION II	20/05	20/05
SHINY APPLICATION	28/05	28/05
Report writing: PEC4	28/05	03/06
PEC4 : PROJECT REPORT	04/06	04/06
Elaboration of the presentation	05/06	11/06
PEC5a : PROJECT PRESENTATION	12/06	12/06
PEC5b : PUBLIC PRESENTATION	13/06	25/06

Figure 1: Gantt chart with the project schedule.

It appears clear now that the initial objectives were too ambitious with such a constrained schedule for someone who has no experience with clustering algorithms nor script writing before starting this master's courses. Nevertheless, the final objective, building a graphical front-end, must be kept in mind, as it has determined the choice of software and algorithms. The development of the graphical front-end is thus considered a future perspective instead of a specific objective of the Final Master's Degree Project.

1.5 Obtained products

1.5.1 Bibliographic review

A bibliographic review has been performed in order to select `R` packages for cytometry data processing, unsupervised clustering algorithms and visualization techniques.

1.5.2 R script to generate synthetic data

A script for synthetic cytometry data generation has been developed in order to benchmark the selected algorithms in controlled conditions.

1.5.3 R scripts for real data pre-processing for algorithm benchmarking

With the finality to use real cytometry data as a reference to evaluate the performance of clustering algorithms, a script has been created. The script allows importing the gating strategies performed on `FlowJo` and labelling the cell examples accordingly.

1.5.4 R scripts for algorithm benchmarking

`R` scripts have been developed allowing to benchmark algorithms with different kinds of data: the generated synthetic data and real samples from 5 and 11-parameter experimental designs. A matching procedure has been developed in order to assign the predicted clusters to the reference populations. Algorithm performances are evaluated in terms of F1 score, number of predicted clusters, number of detected populations and computational time and recorded for further analysis.

1.5.5 Comparative study of clustering algorithms

The results obtained from benchmarking a selection of clustering algorithms are compared and discussed in this report.

1.6 Project structure

The project is structured through its main milestones and the tasks that have been performed to achieve them:

- First selection of unsupervised clustering algorithms based on bibliographic review
- Methods for performance evaluation
- Second selection of unsupervised clustering algorithms based on testing on synthetic data
- Comparative study of selected clustering algorithms
- Discussion
- Conclusions

2 First selection of unsupervised clustering algorithms based on bibliographic review

2.1 Bibliographic research

2.1.1 R packages for cytometry data processing

`Bioconductor` offers several packages performing basic tasks for cytometry data analysis. Table 2 gathers some of the most relevant ones. The package `flowcore` is the main core library providing tools for flow cytometry data management. Most of the `flowCore` functionalities have been tested and discussed by reproducing the examples included in its vignette. Code and comments are shared in the GitHub repository dedicated to this project at https://github.com/plateforme-stlouis/RCyto.

Table 2: R/Bioconductor packages for cytometry data processing

Package	Short description	Ref.
flowCore	Basic Functions for Flow Cytometry Data	17
ggcyto	A ggplot2 graphics implementation	18
flowWorkspace	Infrastructure for representing and interacting with gated and ungated cytometry data sets	19
CytoML	A GatingML Interface for Cross Platform Cytometry Data Sharing	20
flowAI	Automatic and interactive quality control for flow cytometry data	21
flowClean	A quality control tool for flow cytometry data based on compositional data analysis	22

2.1.2 Algorithms for unsupervised learning and visualization methods for multiparametric cytometry data

An extensive bibliographic research on recent high-dimensional cytometry data analysis methods has been performed. The gathered references (that have not all been included in this report) can be consulted in a BibTeX file that is periodically updated and shared at the project's GitHub repository.

The main objective of this project has been selecting unsupervised clustering methods and visualization tools for high-dimensional cytometry data. Recent advances on this field include algorithms, methods and pipelines comprising a broad variety of techniques performing supervised or unsupervised clustering, meta-clustering, automatic partitioning and visualization. Table 3 summarizes the main aspects of a selection of methods performing unsupervised clustering, dimensionality reduction and/or data visualization. The summary is extended on table 4, essentially with unsupervised clustering methods. Table 5 gathers other interesting methods and packages found in the literature that are beyond the scope of this project.

Table 3: Dimensionality reduction, unsupervised clustering and/or visualization methods

Main goal	Method	Language / Environment	Year	Downsampling	Unsupervised learning methods	Number of clusters	Additional methods	Visualization	Comments	Ref.
Dimensionality Reduction	t-SNE	C++, Matlab, Python, R	2008	no	t-SNE	-		t-SNE map (2D scatter plot)		6
Dimensionality Reduction	UMAP	R package	2018	no	UMAP	-		UMAP map (2D scatter plot)		8
Dimensionality Reduction	viSNE	Matlab GUI (CYT), Cytobank online platform	2013	uniform, random	t-SNE	-		t-SNE map (2D scatter plot)		23
Dim. reduction + clustering	Cytosplore +HSNE	Standalone application	2017	no			multi-level hierarchy of non-linear similarities (k-NN) + (BH)-SNE	scatter plots	Highly suitable for the analysis of massive high-dimensional data sets	24
Visualization + clustering	SPADE	R package from GitHub, Matlab, Cytobank online platform	2011	density-dependent	hierarchical, agglomerative clustering	user-defined		MST (dendrogram)	Very high computational time compared to flowSOM (38, 39)	12
Visualization + clustering	ACCENSE	Standalone application with graphical interface	2013	density-dependent	t-SNE, k-means	user specifies a threshold p-value	density-based partitioning	t-SNE map (2D scatter plot)		25
Visualization + clustering	DensVM	R package (cytofkit) from GitHub	2014		t-SNE		density-based partitioning + SVM	t-SNE map (2D scatter plot)		26
Visualization + clustering	FlowSOM	R package from Bioconductor	2015	no	self-organizing map (SOM)	User can define the exact number of clusters or a maximum to try out	consensus hierarchical metaclustering	nodes represented as star charts or pie charts on a 2D grid, MST or t-SNE graphs	Significantly less computational time than SPADE (38, 39)	13
Visualization + clustering	X-Shift	Standalone application (Vortex) with graphical interface	2016				weighted KNN-DE + detection of local maxima + cluster merging (Mahalanobis distance)	Divisive Marker Tree, Force-directed layout		16

Table 4: Unsupervised clustering methods

Main goal	Method	Language / Environment	Year	Unsupervised learning methods	Number of clusters	Additional methods	Visualization	Comments	Ref.
Clustering	flowClust	R package from Bioconductor	2009	multivariate t mixture models with Box-Cox transformation	user-defined or determined by BIC		scatter plots, contour plots, image plots	First clustering step is performed on FSC_H/SSC-H	27
Clustering	flowMerge	R package from Bioconductor	2009	multivariate t mixture models with Box-Cox transformation + cluster merging	user-defined or determined by BIC		scatter plots, contour plots, image plots	Extension of flowClust	28
Clustering	FLAME	GenePattern online platform	2009	multivariate skew t mixture model		2-tiered metaclustering (matching populations across samples)	3D projections, heatmaps		29
Clustering	FLOCK	C source code, Java application	2010	Grid-based partitioning	automatic	density-based partitioning	scatter plots		30
Clustering	SamSPECTRA	R package from Bioconductor	2010	spectral clustering		Faithful Sampling data reduction	scatter plots		31
Clustering	flowMeans	R package from Bioconductor	2011	k-means + merging of clusters	User can also specify a specific or a maximum number of clusters.	change point detection	scatter plots, contour plots, image plots		10
Clustering	flowPeaks	R package from Bioconductor	2012	k-means + finite mixture model	automatic	density-based peak finding	scatter plots		11
Clustering	SWIFT	MATLAB GUI	2014	Gaussian mixture model-based clustering	automatic	unimodal splitting and merging	histograms, scatter plots		32
Clustering	ImmunoClust	R package from Bioconductor	2015	finite mixture model-based iterative clustering		metaclustering	scatter plots		33
Clustering	PhenoGraph	Python, MATLAB GUI (CYT), R package (RPhenoGraph) from GitHub, R package (cytofkit) from GitHub	2015		automatic	k-NNG + Louvain graph partition algorithm			14
Clustering	densityCut	R code available on Bitbucket	2016			k-NNG + local-maxima based clustering			34
Clustering	ClusterX	R package (cytofkit) from GitHub	2016		automatic	density-based partitioning	t-SNE map (2D scatter plot)		9

Table 5: Other approaches and purposes

Main goal	Method	Language / Environment	Year	Unsupervised learning methods	Number of clusters	Additional methods	Visualization	Comments	Ref.
Supervised approach	flowType/Rchy	R package from Bioconductor	2012						35
Supervised approach	flowDensity	R package from Bioconductor	2015			density-based partitioning		Automated gating approach that emulates an expert's sequential 2D gating strategy	36
Supervised approach	GateFinder	R package from Bioconductor	2018					Enriches high-dimensional cell types with simple, stepwise polygon gates requiring only two markers at a time	37
Statistical analysis	Citrus	R package from GitHub, GUI	2014	hierarchical, agglomerative clustering	user defines a minimum cluster size threshold	SAM correaltive model used to identify traits associated with an experimental endpoint	radial hierarchy trees		38
Relationships between categories	oneSENSE	R package from Bioconductor	2015					One-SENSE measures cellular parameters assigned to manually predefined categories, and a one-dimensional map is constructed for each category using t-SNE	39
Multiple time points comparison	FLOW-MAP	R, igraph package, Gephi software package	2015				highly connected graph structure	Compares clusters defined by other algoritnms such as SPADE	40
Joint cell population identification in many flow cytometry samples	BayesFlow	Python, available at GitHub	2016	multivariate Gaussian mixture model					41
Statistical analysis	cytofast	R package from Bioconductor	2018				heatmaps, scatter plots, t-SNE maps, boxplots	Quantification of specific cell clusters and correlations between samples	42
Differences between two groups of samples	CytoBinning	R packages (kernlab, flowCore, e1071, ggplot2)	2018				scatter plots		43
Algorithm benchmarking, cluster comparison	CytoCompare	R package from GitHub	2018						4

2.2 Selection of algorithms

Methods have been selected according the following criteria:

1. Exclude all methods not available on `R`.
2. Include dimensionality reduction methods.
3. Include unsupervised clustering methods.

Informations provided in recent important reviews on the field [1,2] and algorithm benchmarking articles [3,5] have also been considered. Even though the reviews do not perform a proper benchmarking exercise, they give examples of some visualization techniques. Table 6 indicates which methods following our inclusion criteria have been explored in each article.

Table 6: Recent reviews and benchmarking articles on high-dimensional cytometry data analysis

	Mair et al. 2016 [2]	Saeys, Van Gassen, and Lambrecht 2016 [1]	Weber and Robinson 2016 [3]	Kimball et al. 2018 [5]
Type of article	Review	Review	Benchmarking article	Benchmarking article
Method				
SPADE	✓	✓	✓	✓
t-SNE	✓	✓		
PhenoGraph	✓		✓	✓
FlowSOM		✓	✓	
ClusterX			✓	
flowClust			✓	
flowMeans			✓	
flowPeaks			✓	
immunoClust			✓	
SamSPECTRAL			✓	

After considering all this information, the methods detailed below have been selected as the most adequate for the IRSL Core Facilities.

2.2.1 Nonlinear dimensionality reduction techniques

Dimensionality reduction techniques aim to represent multidimensional data in a lower dimensional space. While linear dimensionality reduction methods, such as principal component analysis (PCA), aim to transform the data in a way that best preserves its variance, non-linear approaches search to represent the similarity found in the high-dimensional space.

Dimensionality reduction methods are used to visualize high-dimensional data in two or three dimensions. Additionally, they can be used as a pre-processing step before applying methods for supervised or unsupervised learning.

1. t-SNE (t-Stochastic Neighbor Embedding)
Developed by van der Maaten in 2008 [6], t-SNE is a widely used technique to illustrate and compare different gating, clustering and partitioning strategies on cytometry data. It is also used as previous step in some clustering methods such as `ClusterX`, `ACCENSE` or `DensVM`.

2. UMAP (Uniform Manifold Approximation and Projection)
A promising new technique proposed last year by McInnes, claiming better structure preservation and computational performances than t-SNE [7]. Becht and colleagues have tested it with single-cell (mass cytometry and RNA sequencing) data obtaining satisfactory results in terms of data visualization and computational times. Nevertheless, to our knowledge, UMAP has not been tested as a previous dimensionality reduction step for clustering algorithms applied to cytometry data.

Furthermore, UMAP and t-SNE embeddings were used to train random forest models to predict `Phenograph` clusters' identities leading to similar results [8].

Both t-SNE and UMAP methods have been used in this project as pre-processing clustering steps as well as visualizing methods post-clustering, easing the visualization and the understanding of the clustering results.

2.2.2 Clustering methods

1. `ClusterX`: A density-based partitioning method
This method is included in the `Bioconductor` package for mass cytometry data analysis named `cytofkit`. Its developers have created **`ClusterX`** from the *Clustering by fast search and find of density peaks* (CFSFDP) algorithm introducing modifications to reduce the computational time thanks to a Split-Apply-Combine strategy and automate the density peak detection by using generalized (extreme Studentized deviate) ESD test. `ClusterX` is able to manage up to 4 dimensions, further off, it can be conducted after a t-SNE dimensionality reduction step [9].

2. Clustering methods based on the k-means algorithm: `flowMeans` and `flowPeaks`
These methods are adapted from the k-means algorithm.

Following the multidimensional clustering, **`flowMeans`** performs a merging step based on the Euclidean or Mahalanobis distances that allows to identify concave cell populations. It also includes a change point detection algorithm to determine the number of subpopulations [10].

flowPeaks applies the k-means algorithm with a large K in order to find small clusters allowing the generation of a smoothed density function using the finite mixture model. The number of clusters is then determined by density-based peak finding [11] .

3. Clustering methods with Minimum Spanning Tree visualization

`SPADE`, developed in 2011 by Qiu and colleagues, is one of the main clustering and data visualization methods used nowadays (see table 6) [12]. Briefly, it performs hierarchical, agglomerative clustering after a density-dependent down-sampling step. Results are visualized on an MST dendrogram.

In 2015, Van Gassen et al. published **flowSOM**, a method performing MSTs similar to those obtained with `SPADE` [13]. The main difference between these methods relies on the clustering algorithm. `flowSOM` uses self-organizing maps (SOM), which do not require any down-sampling step. This might be the reason for `flowSOM` achieving remarkably better performance scores and run times than `SPADE` in all the tests realized by Weber and Robinson [3]. For these reasons, only **flowSOM** will be tested as MST-producing method.

4. PhenoGraph

PhenoGraph is nowadays one of the most used partitioning methods for cytometry data. It was developed for high-dimensional cytometry data by Levine and colleagues in 2015 [14]. It has the ability to partition high-dimensional single-cell data with no need of previous clustering nor dimensionality reduction steps. Briefly, the high-dimensional space is modelled using a weighted nearest-neighbor graph (NNG) that is then partitioned using the Louvain community detection method. Importantly, the NNG is constructed in two iterations, namely using the Jaccard similarity coefficient in the second iteration. This should help distinguish small cellular subsets from noise. Even though in the tests performed by Weber and Robinson to detect rare populations (0.03% and 0.8%) `PhenoGraph` does not achieve particularly good F1 scores (0.498 and 0.229, respectively), Kimball et al. point out that `PhenoGraph` can be an interesting tool to identify novel phenotypic subtypes [3,5].

3 Methods for performance evaluation

3.1 F1 score

In order to give guidance on the newly emerging analysis methods for multidimensional flow cytometry data analysis, the *Flow Cytometry: Critical Assessment of Population Identification Methods* (FlowCAP) consortium has organized four different challenges from 2010 to 2014. In the first challenge (FlowCAP-I) the ability of unsupervised methods to reproduce expert manual gating was evaluated [15]. Methods were compared using the F1 score, a widely used performance measure that combines *precision* and *recall* using the harmonic mean. The F1 score ranges from 0 (worst performance) to 1 (best *precision* and *recall* scores). *Precision*, or *positive predictive value*, is the proportion of predicted positive examples that are truly positive, whereas *recall*, which is also known as *sensitivity* or *true positive rate*, measures the proportion of truly positive examples that are correctly classified. Thus, the F1 score is built from the confusion matrix.

The computation of the F1 score requires comparing the clustering results to a labelled reference. For the synthetic data generated in this study there is no need of gating as the examples have been created according specific phenotypic patterns and are thus already labelled. For the real data, the labels are created by manual gating with `FlowJo`. Afterwards, for both synthetic and real data approaches, it is necessary a procedure to assign predicted clusters to cell labels: the matching procedure.

3.1.1 Matching procedure

In FlowCAP-I, the F1 score was calculated for all the possible combinations of predicted clusters and reference populations. Each population was then assigned to the cluster giving the highest F1 score, i. e., the F1 scores were maximized individually [15]. This method allowed a cluster to be assigned to multiple reference populations, a problem that was solved by Samusik *et al* in 2016: by using the Hungarian assignment algorithm, it was the sum of F1 scores that was maximized, limiting the clusters to be assigned to no more than one population [16].

In this work, two different matching procedures have been considered. In order to illustrate them, the clustering result obtained with `RPhenograph` on one of the synthetic data sets generated for this study will be used as example. The methodology used to generate synthetic data will be explained in detail in Chapter 4.1.

In this example, `RPhenograph` has predicted 18 clusters for a data set containing 50,000 cells divided on 11 populations (labels):

```
(t <- table(clustering, labels))
```

```
          labels
clustering    B   NK   T4   T8 NKT_NN NKT_4 NKT_8   U1   U2   U3   U4
        1     0    0   30    0      0     0     0    9  356    0    0
        2     0    0 2743    1      0     0     0    0    1   12    0
        3    32    0    0    0      1     0     0    0    0    9  975
        4     0    8    0    0    959     4     3    0    0   13    0
        5     1    0  131   26     10     0     0   21    0 3116    9
        6     0    0   11    0      4   217     0    0    0    0    0
        7     0    1    5 8702      0     0    16    1    0   51    1
        8     0    0 3078    0      0     6     0    0    0    0    0
        9     0    0    0   18      8     0   731    0    0    1    0
        10    0    0 3643    0      0     1     0    0    2    0    1
        11    0    0 2985    1      0     1     0    0    5    2    0
```

```
12     0 2953    0    0     15     0     0   18    0    0    0
13     0    0 2668    0      0     7     0    1    5   13    0
14  7445    0    0    0      0     0     0   33    0    0   14
15     0    0 3057    0      1    14     0    0    0    5    0
16    22   38    0    1      2     0     0 2042    5   12    0
17     0    0 1433    1      0     0     0    0    0    8    0
18     0    0 2216    0      0     0     0    0    1    8    0
```

3.1.1.1 First matching procedure

In the first matching procedure, a maximum of one cluster is assigned to one reference population. As the generated synthetic data simulates a simplified flow cytometry data set, the method used to assign clusters to partitions has also been simplified. Instead of maximizing the F1 score, only the number of true positives has been maximized. In other words, the procedure finds the most representative cluster for each population:

```r
# Finding and sorting column maximums:
max_values <- apply(t, 2, max)
(sorted_max_values <- sort(max_values))
```

```
 NKT_4     U2  NKT_8 NKT_NN     U4     U1     NK     U3     T4      B
   217    356    731    959    975   2042   2953   3116   3643   7445
    T8
  8702
```

The maximums will be assigned sequentially. Sorting them from smaller to higher assures that, in the case that more than one population find its maximum on the same cluster, the population with a higher number of assigned examples will prevail.

```r
# Finding the maximum number of each cell type (columns)
# on each cluster (rows):
(m <- apply(t, 2, which.max))
```

```
     B     NK     T4     T8 NKT_NN  NKT_4  NKT_8     U1     U2     U3
    14     12     10      7      4      6      9     16      1      5
    U4
     3
```

For instance, the cluster 10 is assigned to the label T4. On the contrary, the cluster 2 owns 2743 T4 cells that have not been assigned. In this case there are 7 clusters (18 clusters - 11 populations) that have not been assigned. However, all the examples must be assigned to a label in order to compute the confusion matrix. Hence, the label *unknowns* is introduced.

```
# Creating a list with n unknowns
matched_preds <- rep("unknown", length(clustering))

# Replacing the numbers of the clusters by the names of the cell types:
for(i in 1:length(clustering)){
  for(j in 1:length(levels(labels))){  #  number of populations
    # we compare to the sorted maximum values:
    if(clustering[i] == m[names(sorted_max_values)[j]]){
      matched_preds[i] <- names(sorted_max_values)[j]
    }
  }
}

# Factorize matched predictions including the "unknown" level
matched_preds_1 <- factor(matched_preds,
                          levels = c(levels(labels), "unknown"))

summary(matched_preds_1)
```

```
      B      NK      T4      T8  NKT_NN   NKT_4   NKT_8      U1      U2
   7492    2986    3647    8777     987     232     758    2122     395
     U3      U4 unknown
   3314    1017   18273
```

18273 unmatched cells have been classed as *unknowns*.

The **confusion matrix** can now be computed:

```
# Adding the label "unknown" to the labels:
all_labels <- labels
levels(all_labels) <- c(levels(labels), "unknown")

(confusionMatrix(data = matched_preds_1, reference = all_labels))$table
```

```
          Reference
Prediction     B    NK    T4    T8 NKT_NN NKT_4 NKT_8    U1    U2    U3
   B        7445     0     0     0      0     0     0    33     0     0
   NK          0  2953     0     0     15     0     0    18     0     0
   T4          0     0  3643     0      0     1     0     0     2     0
   T8          0     1     5  8702      0     0    16     1     0    51
   NKT_NN      0     8     0     0    959     4     3     0     0    13
   NKT_4       0     0    11     0      4   217     0     0     0     0
   NKT_8       0     0     0    18      8     0   731     0     0     1
   U1         22    38     0     1      2     0     0  2042     5    12
   U2          0     0    30     0      0     0     0     9   356     0
   U3          1     0   131     26      10      0      0     21      0   3116
   U4         32     0     0      0       1      0      0      0      0      9
   unknown     0     0 18180      3       1     28      0      1     12     48
          Reference
Prediction    U4 unknown
   B          14       0
   NK          0       0
   T4          1       0
   T8          1       0
   NKT_NN      0       0
   NKT_4       0       0
   NKT_8       0       0
   U1          0       0
   U2          0       0
   U3          9       0
   U4        975       0
   unknown     0       0
```

In this case, the B cells have been correctly assigned, but most of the T4 cells have been classed as unknowns.

3.1.1.2 Second matching procedure

In both the FowCAP-I challenge and the work of Samusik using the Hungarian assignment algorithm, the objective was to evaluate methods for automatic gating. It made sense performing one-to-one assignments, i. e., assigning one population to only one cluster [15,16]. However, the present study aims to evaluate exploratory procedures making special attention to rare populations. This is no without cost: algorithms finding small populations will tend to split the largest ones. It is therefore assumed that populations can be predicted by more than one cluster and that results will require validation. Therefore, instead of choosing the cluster giving a better F1 score or a greater number of true positives, all the clusters matching for the same population will be merged and the F1 score will be calculated for the merged clusters.

The matching procedure has thus been modified as follows.

1. Each cluster will be assigned to the population for which it contains a higher number of cells. Several clusters can be assigned to the same label. There will be no *unknowns*.

```
# Finding the cell population (columns)
# with a higher number of cells for each cluster (rows):
(m <- apply(t, 1, which.max))
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
 9  3 11  5 10  6  4  3  7  3  3  2  3  1  3  8  3  3
```

Here, the first row indicates the cluster numbering, and the second row is the population label. For example, the label B (1) is assigned to the cluster 14.

In this case, the population 3 (T4) has been split in 8 clusters, as it can be appreciated in the following table.

```
table(m)
```

```
m
 1  2  3  4  5  6  7  8  9 10 11 
 1  1  8  1  1  1  1  1  1  1  1 
```

In this case, population 3 (T4) has been split in 8 clusters.

2. Matching cell labels and clusters:

```
# Empty list
matched_preds <- rep("NA", length(clustering))

# Replacing the numbers of the clusters by the names of the cell types:
for(i in 1:length(clustering)){
  for(j in 1:length(m)){ # Number of predicted clusters
    if(clustering[i] == names(m)[j]){
      matched_preds[i] <- levels(labels)[m[[j]]]
    }
  }
}

# Factorize matched predictions
matched_preds_2 <- factor(matched_preds, levels = levels(labels))
```

3. Computing the confusion matrix:

```
(confusionMatrix(data = matched_preds_2, reference = labels))$table
```

```
          Reference
Prediction     B    NK    T4    T8 NKT_NN NKT_4 NKT_8    U1    U2    U3
    B       7445     0     0     0      0     0     0    33     0     0
    NK         0  2953     0     0     15     0     0    18     0     0
    T4         0     0 21823     3      1    29     0     1    14    48
    T8         0     1     5  8702      0     0    16     1     0    51
    NKT_NN     0     8     0     0    959     4     3     0     0    13
    NKT_4      0     0    11     0      4   217     0     0     0     0
    NKT_8      0     0     0    18      8     0   731     0     0     1
    U1          22    38     0      1       2      0      0  2042      5     12
    U2           0     0    30      0       0      0      0     9    356      0
    U3           1     0   131     26      10      0      0    21      0   3116
    U4          32     0     0      0       1      0      0     0      0      9
          Reference
Prediction    U4
    B         14
    NK         0
    T4         1
    T8         1
    NKT_NN     0
    NKT_4      0
    NKT_8      0
    U1         0
    U2         0
    U3         9
    U4       975
```

The 18180 T4 cells previously classed as *unknowns* by the first matching procedure are now included in the partition matched to the T4 cells label ($3643 + 18180 = 21823$).

3.1.1.3 Visualization

In order to visualize labels, predictions and matching results with single cell resolution, data (which is five-dimensional) is reduced to two dimensions using the t-SNE algorithm and mapped onto scatter plots (Figure 2). Original cell labels (ground truth), predicted clusters and partitions resulting from the two matching procedures are indicated. It can be observed that the first matching procedure (bottom left) classes most of the T4 cells (upper left, yellow) as *unknowns* (pink).

3.1.2 Mean F1

The F1 score for each population has been calculated using the partitions matched with the second procedure. Afterwards, the F1 scores will be averaged in order to obtain a global performance score.

In the FlowCAP-I study, the F1 scores are normalized by the size of the population. The sum of the normalized scores produces an F1 measure. As the size of the populations is referred to all the reference populations, whether they have been detected by the clustering algorithm or not, the final F1 measure computation penalizes the fact of missing populations [15]. In the benchmarking article from Weber and Robinson, the F1 scores are averaged without weights, considering equally important small and large populations [3]. As the present study aims to

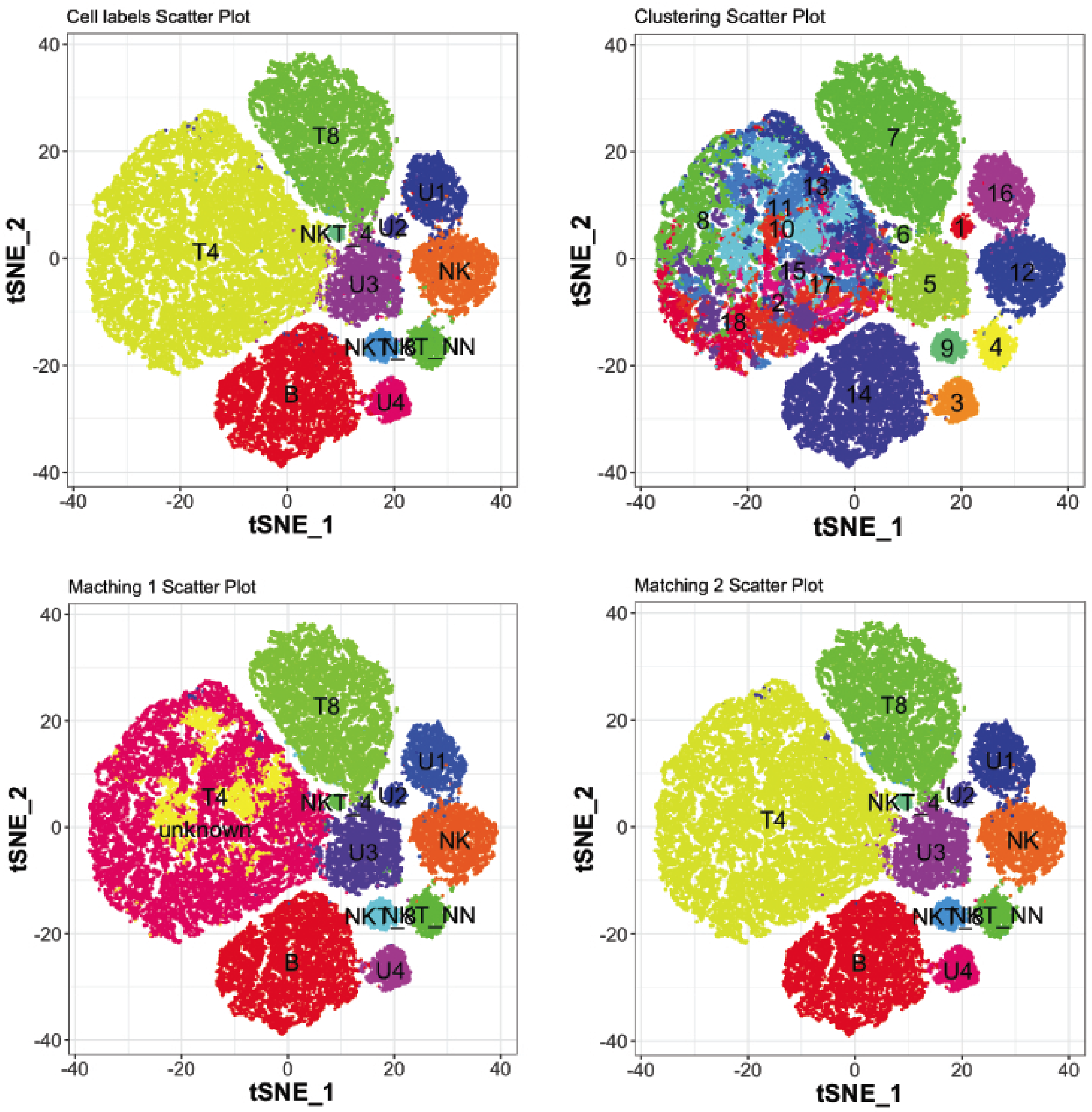


Figure 2: t-SNE mapping of the original five-dimensional data. Data points are colored according reference cell labels (upper left), the RPhenograph clustering (upper right), and the partitions resulting from the two matching procedures (bottom).

select algorithms particularly able to detect rare populations, the mean F1 scores have also been computed unweighted.

3.1.2.1 Fewer clusters than populations

In the first algorithm benchmarking approach using synthetic data, the fact of missing populations has been penalized as in the FlowCAP-I study by according the undetected reference populations with an F1 score of zero. On the contrary, for the subsequent benchmarking on real cytometry data, only the detected populations have been taken into consideration to compute the mean F1 score. This change of criterium is due to the fact that with the synthetic data, the algorithms that accept specifying number of clusters as a parameter have been run asked to find the exact number of reference populations, whereas with the real data it has been analyzed the fact of performing a clustering asking to find fewer clusters than reference populations. Besides, the tests performed with real data have been analyzed more in detail, paying attention to the number of clusters detected. The mean F1 score computed for the real data tests, thus, does not indicate the global algorithms' ability to identify the different populations, but the overall degree of accuracy for the detected populations.

3.1.2.2 Outliers

The synthetic samples produced in this project do not contain any outlier or unknown cell: all the examples are assigned to a cell label. However, manually gating real data can produce outliers, including extremely different examples: cells with atypical marker expressions, intermediate forms or experience artefacts. Thus, algorithms are not expected to detect a common pattern in this group of examples. Nevertheless, outliers should not be excluded from the data sets in order to test algorithm performances in real conditions. Therefore, the outliers have been maintained to feed the clustering algorithms but excluded before building the confusion matrix.

3.2 CPU time

Running *user* times have been recorded as a measure of computational complexity. All the tests have been performed in the same machine, a MacBook Pro with an Intel Core i7 processor and 16 GB of RAM.

4 Second selection of unsupervised clustering algorithms based on testing on synthetic data

4.1 Clustering algorithms

The algorithms selected based on the bibliographic review are listed in Table 7, indicating whether they have been used with high-dimensional and/or dimensionality-reduced data, and whether there is a possibility to indicate the number of desired clusters.

`ClusterX` cannot be applied to data with more than four dimensions, thus, a dimensionality reduction step is required. `FlowSOM` and `RPhenograph` algorithms are designed to cluster high-dimensional data, the former by building self-organizing maps and the latter by measuring distances in a k-Nearest Neighbor graph. It does not seem useful to reduce the data before applying these algorithms. The methods based on the k-means algorithm, `flowMeans` and `flowPeaks`, do not need any previous dimensionality reduction step neither, but have been used to test whether non-linear dimensionality reduction is useful to improve their clustering performances. Two different techniques for dimensionality reduction have been tested: t-SNE and UMAP.

Table 7: First selected clustering algorithms

	Method	High-dimensional data	Dimensionality-reduced data	Number of clusters
1	FlowSOM	✓		User-defined
2	RPhenograph	✓		Cannot be specified
3	flowMeans	✓	✓	User can indicate a maximum number of clusters
4	flowPeaks	✓	✓	Cannot be specified
5	ClusterX		✓	Cannot be specified

4.2 R script to generate synthetic data

An `R Markdown` script has been developed in order to generate controlled synthetic data. Each run produces a sample with 11 populations defined by 5 markers according to real data. Marker expression values follow normal distributions.

The following chapters summarize the main aspects of the script, that is available on the GitHub repository dedicated to this project.

4.2.1 Parameters

The script takes as parameters:

- The name of the synthetic sample
- The number of cells
- A random seed

Additional information can be modified in the first chunk:

- Mean and SD for the negative ("low") markers
- Mean and SD for the positive ("high") markers
- Table of phenotypes
- Percentage of cells per population. This information is exported to be used to evaluate algorithm performances.

A phenotype table is built indicating the expression level pattern for each cell type, as in this example:

```r
B <- c("CD3" = "low",  "CD4" = "low",  "CD8" = "low",
       "CD56" = "low", "CD19" = "high")
```

4.2.2 Sample generation

The procedure is based in three steps:

- Values are generated according the phenotype table using the `rnorm()` function in order to follow a normal distribution.
- Synthetic cell values are randomly rearranged in order to simulate a real sample.
- Cells labels are exported in order to be used to evaluate algorithm performances.

Below is shown the core of the script:

```r
# creating the dataframe
set.seed(params$seed)
for(i in (1:length(cells))){  # i takes values [1, number of CELL TYPES]
  # For cell type i, repeat its name as many times as the number of cells
  c <- rep(cells[i], ns[i])
  # One column will be created for every marker with intensity values
  for(j in (1:length(markers))){
    if(pheno[i,j] == "low"){
      p <- rnorm(ns[[i]], mean = neg_mean, sd = neg_sd)
    }else{
      p <- rnorm(ns[[i]], mean = pos_mean, sd = pos_sd)
    }
    # all the columns (one per marker) are joined:
    # convert the "c" list as data frame
    # to preserve numeric "p" values after cbind-ing
```

```
      c <- cbind(as.data.frame(c), p)
    }
    if(i==1){
      # Values for the first cell type start to fill the dataframe "pop"
      pop <- c
    }else{
      # Values for the other cell types are joined to "pop"
      pop <- rbind(pop, c)
    }
}

# Name the variables:
names(pop) <- c("cells", markers)


#### REARRANGING
# Randomly sampling
set.seed(42)
x <- sample(1:n, n, replace = F)

pop <- pop[x,] # Rearranging the dataframe

rownames(pop) <- 1:n # Renaming the rows (in order)


# Save the column with the cell labels to a file
saveRDS(pop$cells,
        paste("labels", params$pop_number, Sys.Date(), sep = "_"))

head(pop)
```

```
   cells      CD3       CD4        CD8      CD56      CD19
1     U2  508.589  4962.590  928.24123 -208.6189 -579.0167
2     U3 3452.062 -2542.275 -729.86929 -181.6612 -181.8577
3     T4 4152.289  5555.634  714.79808 -227.0640 -734.4824
4 NKT_NN 5818.124 -1460.082 1104.66106 6156.4867  330.6812
5     T4 4398.932  4267.739 -639.52833  522.0184  204.6559
6     T4 3223.760  4304.875  -86.19721 -747.4541 2552.5026
```

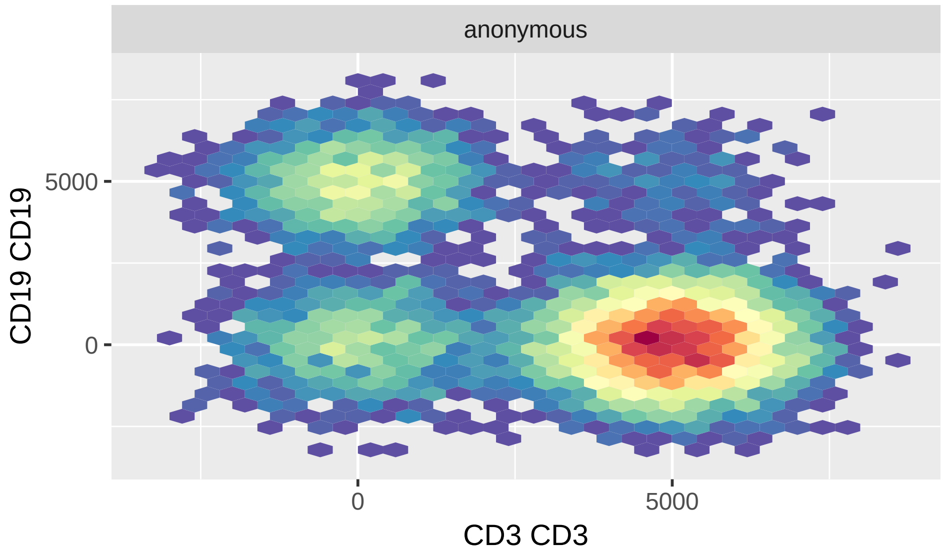
4.2.3 Conversion to FlowFrame class objects

The data frame is stored as an object of class `flowFrame`. This is the format used by `flowCore` to manage FCS (Flow Cytometry Standard) files.

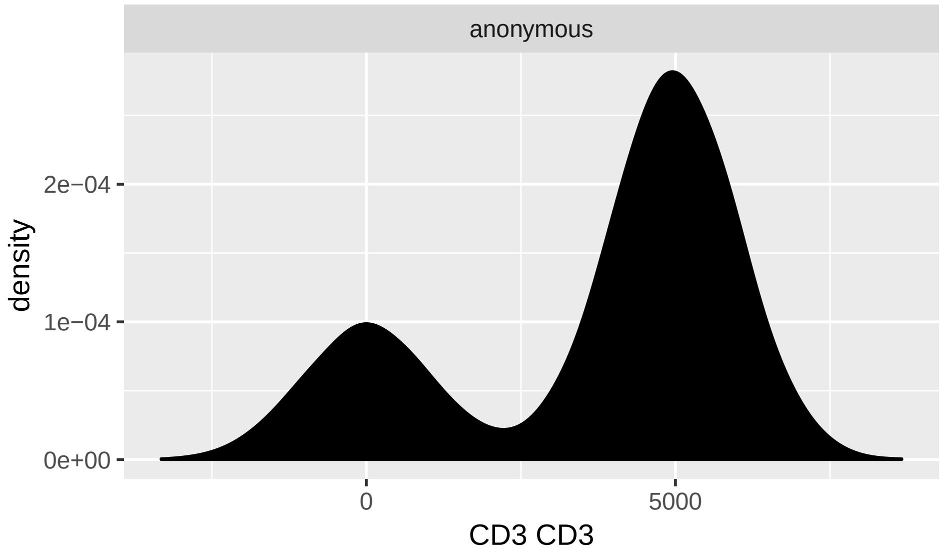
```r
library(flowCore)
ff <- new("flowFrame", exprs = as.matrix(pop[,-1]))
```

Marker expression can be visualized using the tools provided by the package `ggcyto` for cytometry data:

```r
library(ggcyto)
autoplot(ff, markers[1], markers[5])
```



```r
autoplot(ff, markers[1])
```



4.2.4 Export to .fcs

Finally, a synthetic FCS data file is exported.

```r
write.FCS(ff,
          paste("ff_", params$pop_number, "_", Sys.Date(), ".fcs", sep = ""))
```

4.3 Synthetic data generated for algorithm benchmarking

Eight synthetic samples have been generated according to the following parameters:

- 11 populations
- 5 markers
- Mean for negative (low) markers: 50
- Mean for positive (high) markers: 5000

The 11 phenotypes are created according the expression patterns indicated in Table 8.

Table 8: Phenotypes

	CD3	CD4	CD8	CD56	CD19
B	low	low	low	low	high
NK	low	low	low	high	low
T4	high	high	low	low	low
T8	high	low	high	low	low
NKT_NN	high	low	low	high	low
NKT_4	high	high	low	high	low
NKT_8	high	low	high	high	low
U1	low	low	low	low	low
U2	low	high	low	low	low
U3	high	low	low	low	low
U4	high	low	low	low	high

Cell number, standard deviations and cell type proportions have taken different values in function of the conditions indicated in Table 9.

Table 9: Parameters used for synthetic sample generation

	SD	percenteages	n
Condition 1	500	equal	10000
Condition 2	1000	equal	10000
Condition 3	1000	different	10000
Condition 4	1000	different	50000

In conditions 3 and 4, the generated samples are composed of populations in different percentages according a real sample, as indicated in Table 10.

Table 10: Cell populations percentages

	B	NK	T4	T8	NKT_NN	NKT_4	NKT_8	U1	U2	U3	U4
%	15	6	44	17.5	2	0.5	1.5	4.25	0.75	6.5	2

Each condition has been used to generate two different samples by using two different random seeds, resulting in 8 final samples. Samples are identified with the number of the condition and the letters “A” or “B” for the different random seeds. For example, sample “2 B” has been generated with condition 2 and second seed (B).

4.4 Algorithm benchmarking script

An `Rmd` script has been written and used to test the selected methods with the generated samples. The same script includes all the methods and is run one at a time for every sample. The script is available at the project repository. The main aspects of the procedure are indicated below.

4.4.1 Random seeds

Algorithms requiring a random start (`FlowSOM`, t-SNE and UMAP) have been tested five times with different random seeds in order to test their reproducibility.

4.4.2 Number of clusters

All the synthetic samples produced to benchmark the clustering algorithms contain 11 populations. `FlowSOM` does not determine automatically the number of clusters; it must be specified by user. There is also the option of choosing a maximum number of clusters to be tested, but it is not recommended [3]. The `flowMeans` method determines the number of clusters automatically, but the user has the option to indicate a maximum number of clusters. These algorithms have been run asked to find exactly or a maximum of 11 clusters, respectively. `flowMeans` has always produced the maximum number of clusters indicated with the tested synthetic samples (Figures 5 and 8).

4.4.3 Functions

The matching procedure and the computation of the mean F1 score are performed through functions according the methodology discussed in Chapter 3.

4.4.3.1 Matching procedure

This procedure allows merging the clusters that match to the same reference population into the same partition.

```r
matching <- function(prediction, labels){
  # cross table
  t <- table(prediction, labels)

  # Finding the cell population (columns)
  # with a higher number of cells for each cluster (rows):
  m <- apply(t, 1, which.max)

  # Empty list
  matched_preds <- rep("NA", length(prediction))

  # Replacing the numbers of the clusters by the names of the cell types:
  for(i in 1:length(prediction)){
    for(j in 1:length(m)){ # Number of predicted clusters
      if(prediction[i] == names(m)[j]){
      matched_preds[i] <- levels(labels)[m[[j]]]
      }
    }
  }

  # Factorize matched predictions
  matched_preds <- factor(matched_preds, levels = levels(labels))

  matched <- list("preds" = matched_preds, "m" = m, "t" = t)
  return(matched)
}
```

4.4.3.2 Computing the mean F1

In this case, the F1 scores for the undetected populations are fixed at zero and taken into account for the F1 average.

```r
mean_f1 <- function(cm, c){
  # Extracting the F1 values
  f1_list <- cm$byClass[,"F1"]
  # replacing NAs by zero
  f1_list[is.na(f1_list)] <-  0

  return(mean(f1_list[1:c]))
}
```

4.4.4 Clustering

Each method is performed using the `matching()` and `mean_f1()` functions as in this example:

```
pred <- flowMeans(ff@exprs, MaxN = c )

# Macth labels and predictions (cell level)
matched <- matching(pred@Labels[[1]], labels)

# Storing the results
flowmeans_pred <- matched$preds

# COMPUTING THE CONFUSION MATRIX AND OTHER PERFORMANCE MEASUREMENTS
# store and print
print(matched$t)
(flowmeans_cm <-
    confusionMatrix(data = matched$preds, reference = labels))$table

# COMPUTE THE MEAN F1
flowmeans_F1 <- mean_f1(flowmeans_cm, c)
```

The intermediate cross tables and the confusion matrices are printed for the `html` documents generated with the `Knit` package. The confusion matrix elements produced by the `confusionMatrix()` function from the package `caret` (that include the F1 scores for all the populations) and the computed mean F1 scores are recorded for further analyses.

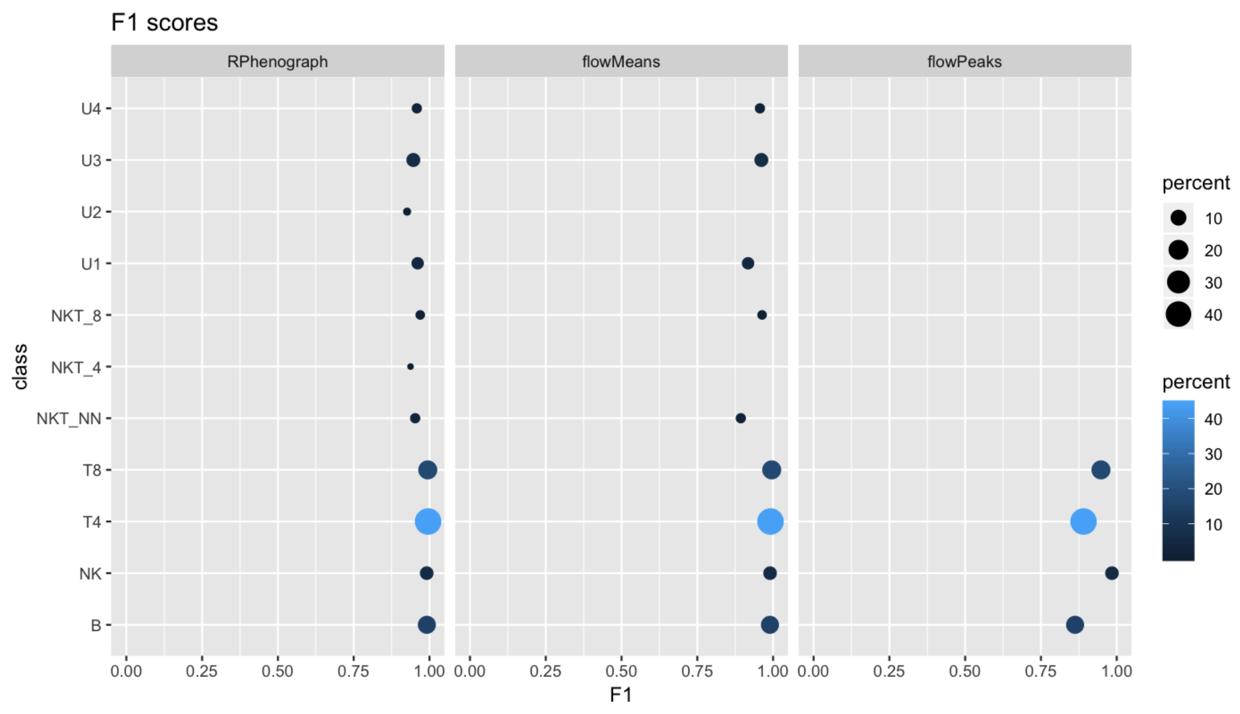


Figure 3: F1 scores obtained for sample "3 A" for all the cell types using RPhenograph, flowMeans or flowPeaks. Dot sizes and coloring both indicate populations' frequencies. Missing points on a row indicate undetected populations.

4.5 Evaluation of performance

4.5.1 Methods without random start

The clustering algorithms `RPhenograph`, `flowMeans` and `flowPeaks` do not need any random start. Thus, each method has been tested for each sample only once. Figure 3 shows the F1 scores computed for a sample generated with the condition 3 (sample "3 A") using these methods. `RPhenograph` makes good predictions for all the populations. `flowMeans` fails to detect the smallest populations (U2 and NKT_4). Finally, `flowPeaks` fails to detect all populations below 6%.

This analysis has been performed for the four synthetic sample types in duplicate. The mean F1 has been computed for every method and sample. This enables performing a global comparison but lacks the detail for the specific populations.

On Figure 4.5.1, mean F1 values obtained with the methods without random start are compared. It can be appreciated that changing the cell type proportions (conditions 3 and 4) reduces dramatically the global performances. `RPhenograph` is the only method that keeps producing good scores in these circumstances. Thus, `flowMeans` and `flowPeaks` methods (without previous dimensionality reduction) will be discarded.

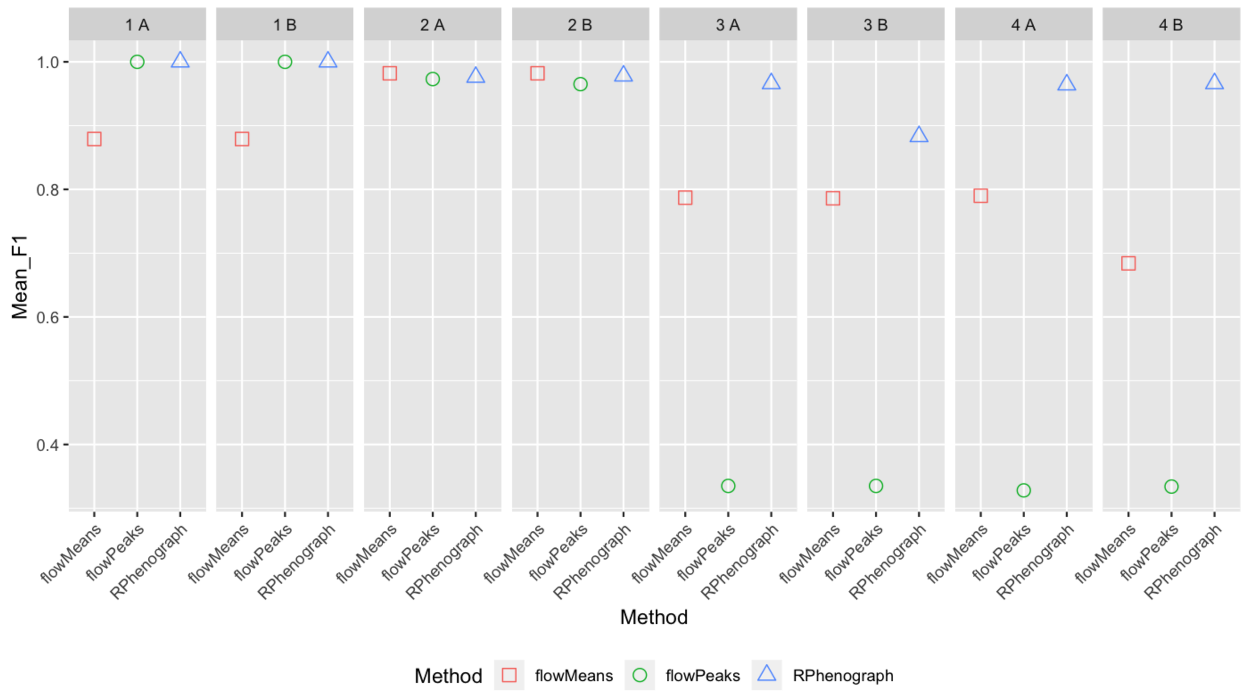


Figure 4: Mean F1 scores for all the samples generated using RPhenograph, flowMeans or flowPeaks.

Figure 5 gives important additional information about `RPhenograph` performances. In this graph, mean F1 scores are related to the number of detected clusters. For `flowMeans` it has been specified to limit the number of clusters to 11. As it happened with sample "3 A" (Figure 3), `flowPeaks` only predicts 4 clusters for the samples generated according conditions 3 and 4, drastically decreasing the F1 score (Figure . `RPhenograph` achieves good F1 scores, but it can be seen here that it is not without cost: increasing the samples complexity (conditions 3 and 4) also increases the number of detected clusters (15-19).

4.5.2 Methods with random start

`FlowSOM` clustering method and dimensionality reduction techniques t-SNE and UMAP use a random start. The F1 scores obtained with these methods are plotted as box plots in order to see their reproducibility. Five different random starts have been used.

Figure 6 shows the results obtained with the sample "3 A". A huge dispersion is observed with the methods using `flowMeans`, meaning that this partitioning method (that does not use any random start) would be extremely sensitive to changes on the t-SNE and UMAP maps. On the contrary, `flowPeaks` partitioning is highly reproducible after UMAP dimensionality reduction. Nevertheless, this algorithm fails to detect the rarest populations (U2 and NKT_4), whether it is performed following t-SNE or UMAP dimensionality reduction. `ClusterX` algorithm is also very reproducible. It fails to detect the NKT_4 population when it is applied following

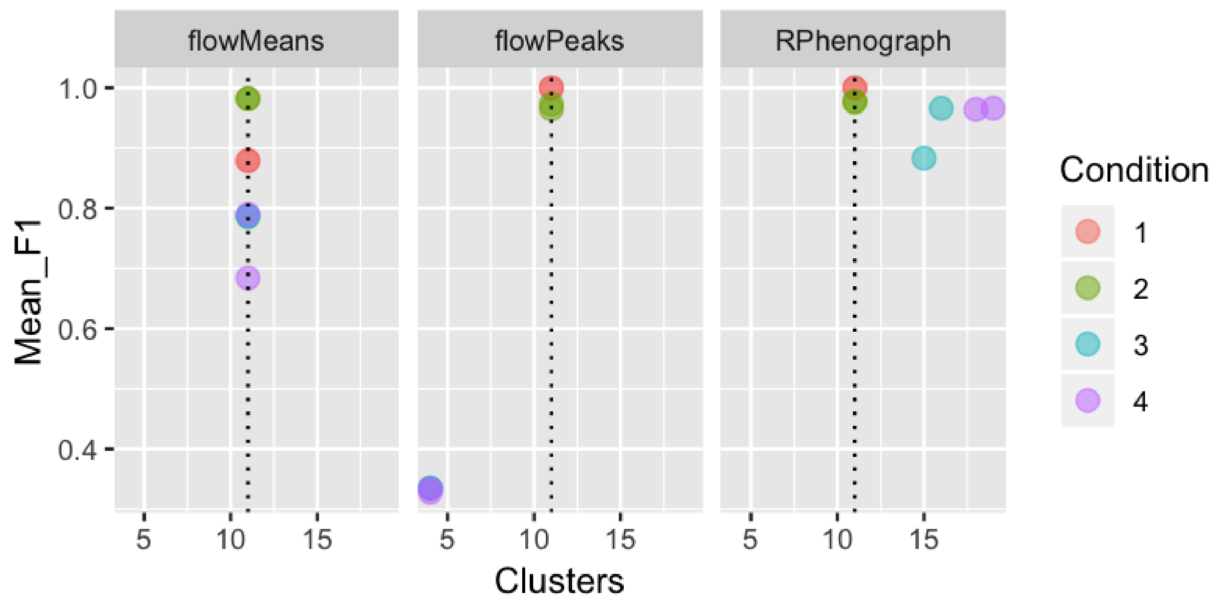


Figure 5: Mean F1 scores for all the samples (4 conditions performed in duplicate) related to the number of predicted clusters obtained using RPhenograph, flowMeans or flowPeaks. Dotted lines indicate the number of populations (ground truth).

t-SNE but finds all the populations after UMAP dimensionality reduction. `FlowSOM` also manages to detect all the cell populations, but with some more variability on the small populations (<6%).

Figure 7 shows the mean F1 values obtained for the random start methods with five different seeds. Again, increasing the sample complexity by including populations with extremely different frequencies decreases the overall performances. `flowMeans` is definitely giving poor results for these samples and will thus be discarded. On the other hand, `flowPeaks` is potentially interesting, particularly after UMAP dimensionality reduction, as it gets improved when the total number of cells increases (condition 4 *vs* condition 3). In fact, condition 4 samples contain 50,000 cell examples; it is possible that this algorithm performs better with samples having a higher number of cells, which is often the case. On the other hand, `ClusterX` does not seem to be affected by the sample size and tends to give better performances when it is conducted after UMAP dimensionality reduction. In fact, UMAP + `ClusterX` method achieves equivalent or even better performances than `FlowSOM`, an algorithm that generates good mean F1 values in all conditions.

Finally, graphs on Figure 8 give information about the number of predicted clusters, which has been fixed to 11 in `FlowSOM` and to a maximum of 11 in `flowMeans`. It can be observed that with `flowPeaks`, both F1 scores and the number of predicted clusters get improved as the sample size increases. Following UMAP, `flowPeaks` produces better and more reproductible predictions. `ClusterX` performs clearly better following UMAP. As it can be seen, following t-SNE, `ClusterX` produces similar mean F1 scores for conditions 3 and 4 samples. Nevertheless,

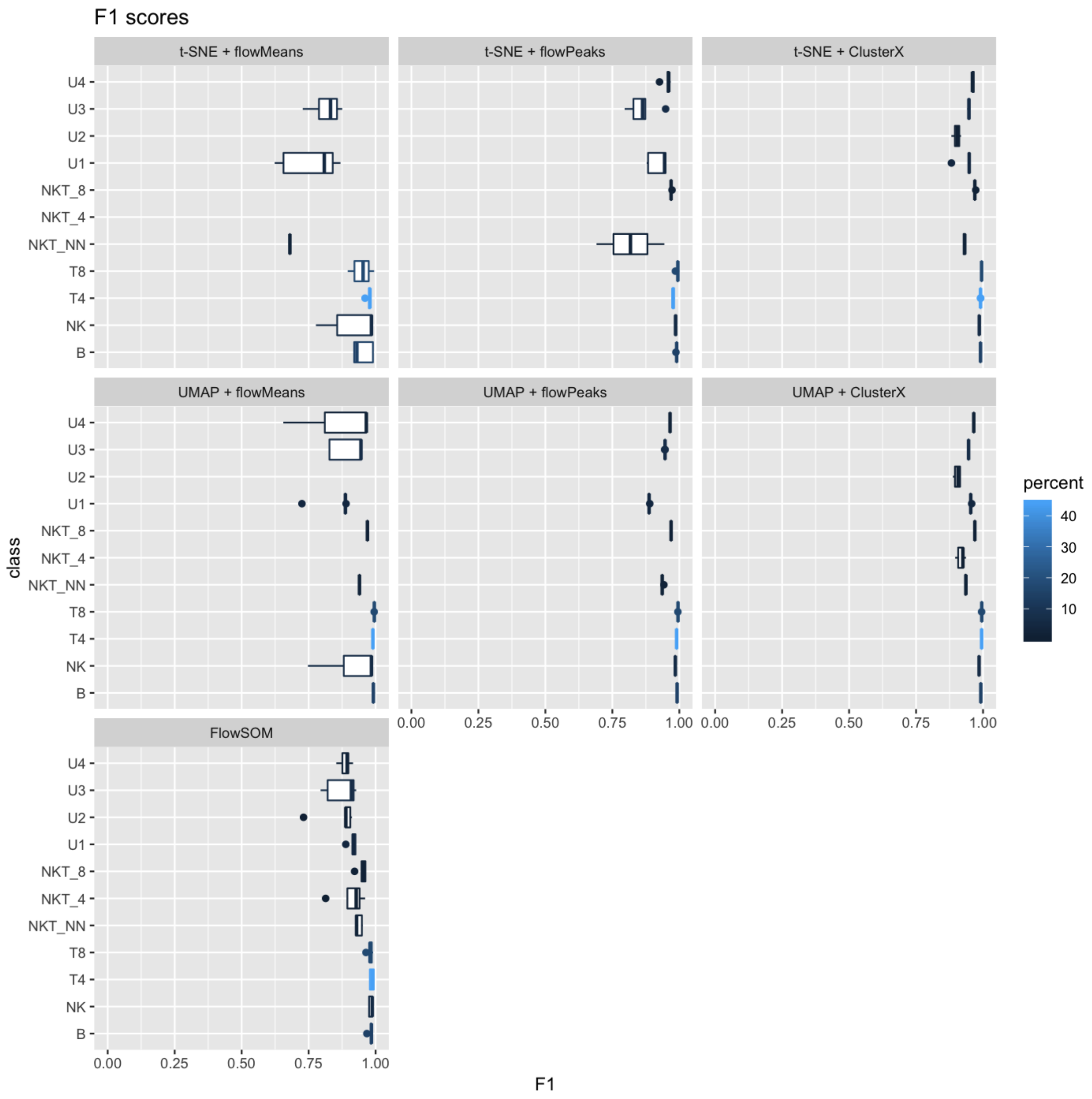


Figure 6: F1 scores obtained for sample "3 A" for all the cell types using flowSOM and the dimensionality reduction algorithms t-SNE and UMAP followed by flowMeans or flowPeaks clustering methods. Coloring indicates the populations' frequencies. n = 5 random starts for every method.

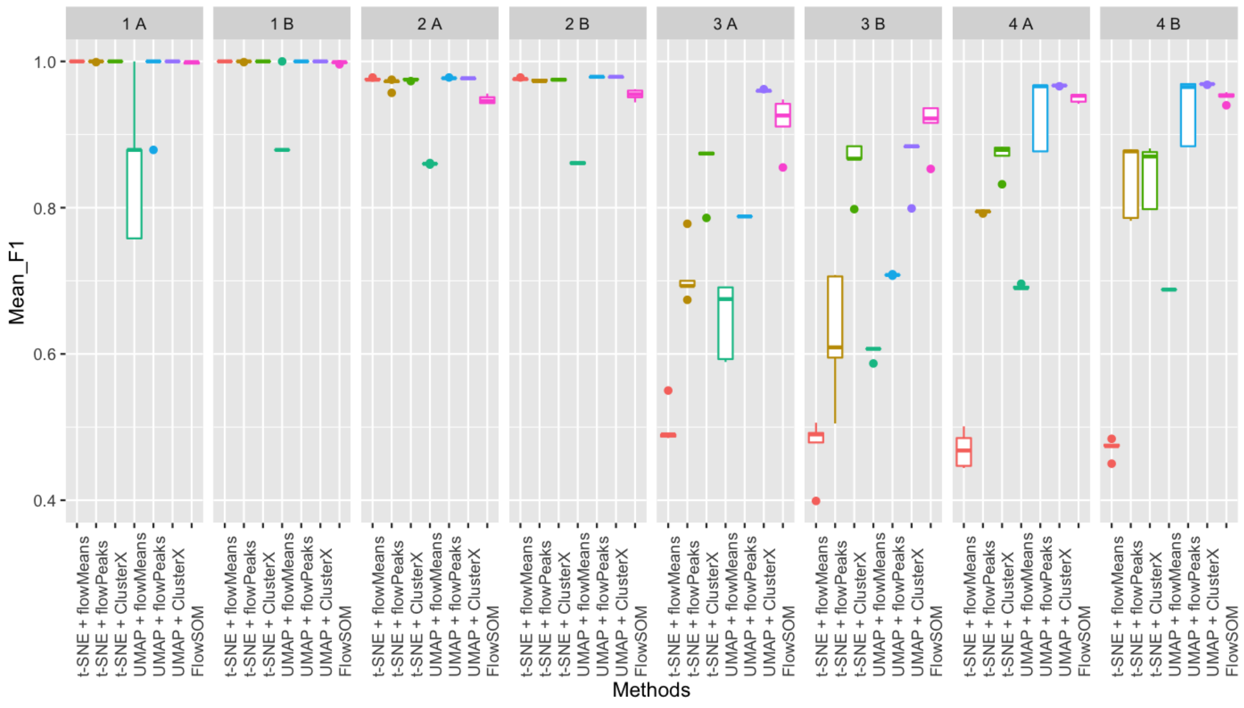


Figure 7: Mean F1 scores for all the samples generated using flowSOM and the dimensionality reduction algorithms t-SNE and UMAP followed by flowMeans, flowPeaks or ClusterX clustering methods. n = 5 random starts for every method.

increasing the sample size (condition 4) dramatically increases the number of predicted clusters. On the contrary, when `ClusterX` is conducted following UMAP dimensionality reduction, the number of predicted clusters is closer to the ground truth and is not affected by the sample size. Moreover, better mean F1 scores are achieved with the largest samples.

4.6 Conclusions

Synthetic samples have been evaluated using a matching procedure that does not penalize the splitting of populations and taking into consideration the number of predicted clusters. The following methods have been selected to be tested with real data:

- `flowSOM`
- `RPhenograph`
- UMAP + `flowPeaks`
- UMAP + `ClusterX`

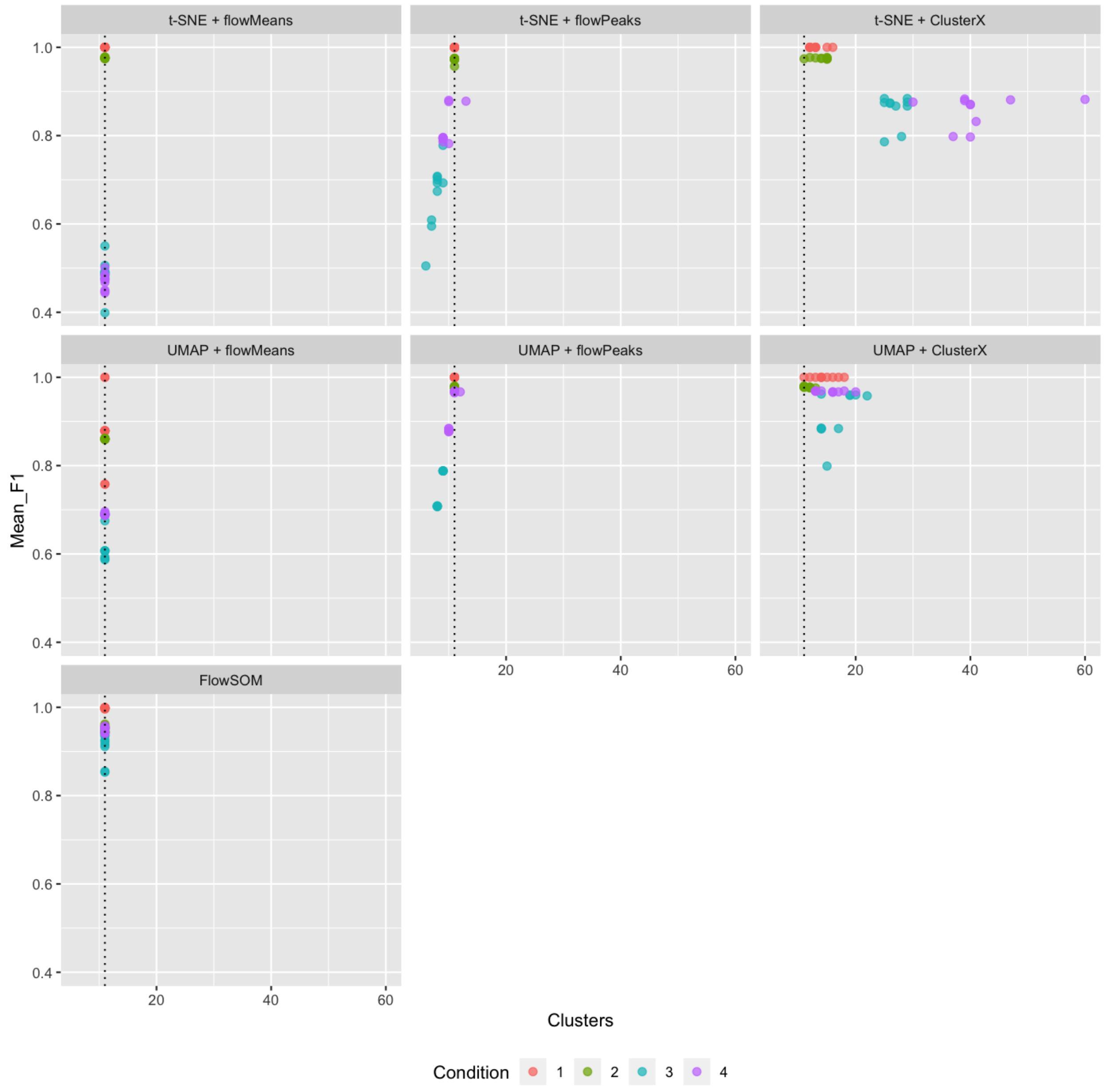


Figure 8: Mean F1 scores for all the samples (4 conditions performed in duplicate) related to the number of predicted clusters obtained using flowSOM and the dimensionality reduction algorithms t-SNE and UMAP followed by flowMeans, flowPeaks or ClusterX clustering methods. n = 5 random starts for every method. Dotted lines indicate the number of populations (ground truth).

5 Comparative study of selected clustering algorithms

The clustering algorithms selected after being tested on synthetic data are applied to real flow cytometry samples generated at the IRSL Flow Cytometry facility according two experimental designs aiming to characterize human lymphocytes. The first design uses five parameters to characterize five main populations. The second experimental design counts with 11 parameters allowing for the characterization of several subpopulations from the same main populations defined with the 5-parameter design. Actually, some other parameters have been used to define the ensemble of single-cell, living lymphocytes, such as the Forward Scatter (FSC) or the Side Scatter (SSC). The 5 or 11 parameters are the ones used to explore this particular cell subset.

5.1 5-parameter design

Data is manually gated with `FlowJO` as depicted in Figure 9. Five main populations are defined: T, NKT, NK, B and NO_BTNK (no B, no T, no NK). Moreover, 20 additional populations have been defined according to the different expression patterns that can be distinguished with the five markers used.

5.1.1 Data preparation

An `R Markdown` script has been developed in order to prepare flow cytometry data for algorithm benchmarking. The script enables importing and exploring the data, applying the hierarchical strategy performed on `FlowJo`, labelling the examples according to this gating strategy, performing down-sampling and exporting the data for further analysis. The whole script is available at the GitHub repository. Its main aspects are indicated below:

5.1.1.1 Importing data from FlowJo workspace

The `FlowJo` workspace (`.wsp`) holding the gating strategy is open using the `openWorkspace()` function from the `flowWorkspace` package. The workspace is then parsed with the corresponding `.fcs` files through the `parseWorkspace()` function, creating a `GatingSet` object.

```r
library(flowWorkspace)
ws <-  openWorkspace(params$wsp)
gs <-  parseWorkspace(ws, name = 2, sampNloc="sampleNode")
# 1: All Samples, 2: group
```

The gating strategy can now be visualized (see Figure 10), showing the gating performed on `FlowJo` (Figure 9).

The expression data can be extracted from the `GatingSet` object:

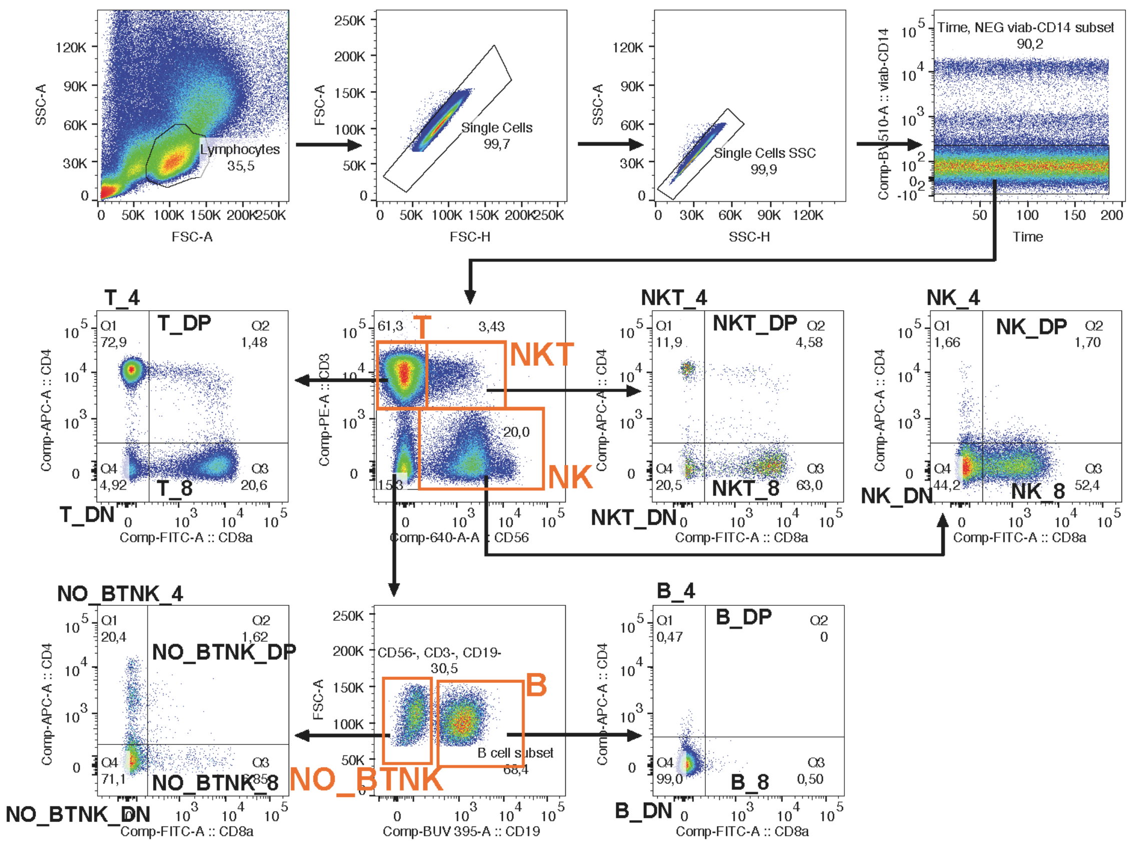


Figure 9: Manual gating strategy for the 5-parameter experimental design. The five main populations are indicated in orange.

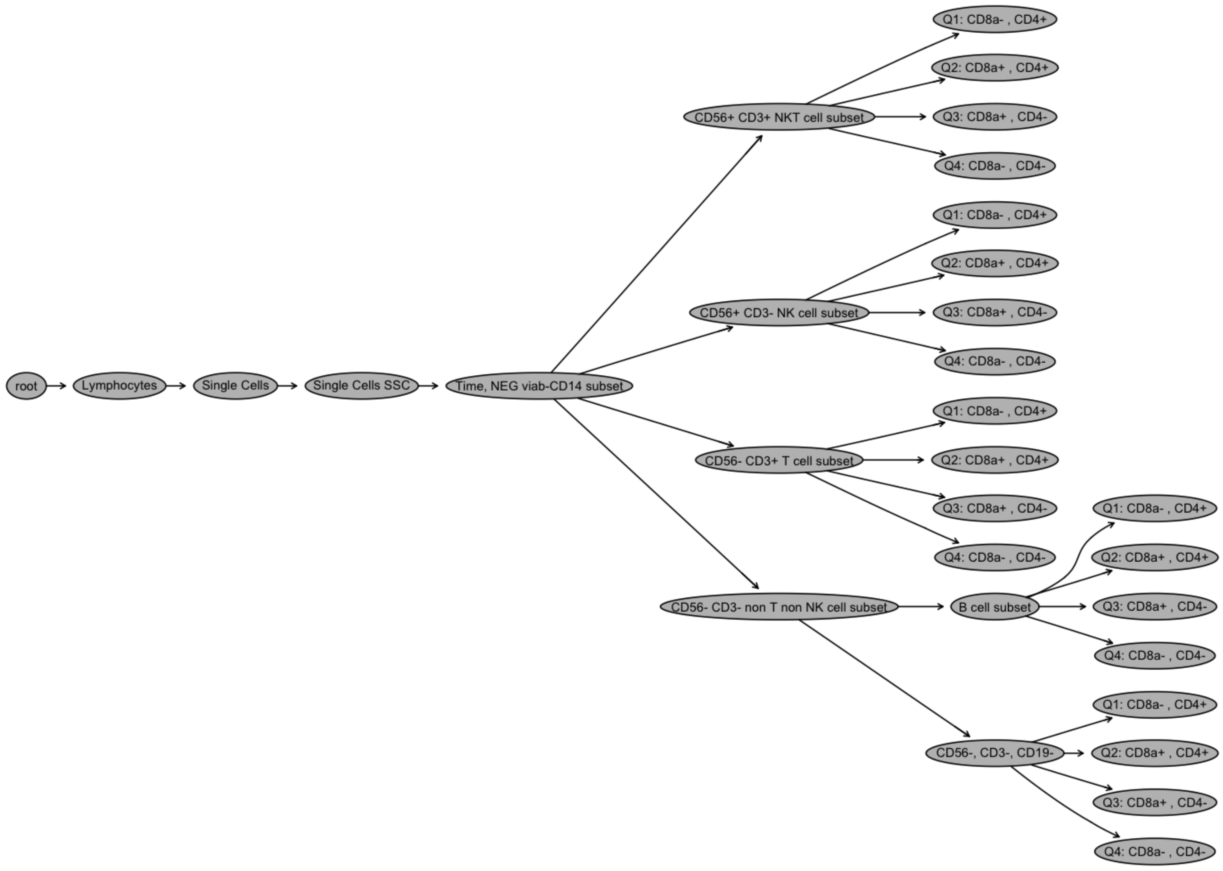


Figure 10: Representation of the hierarchical gating strategy for the 5-parameter design imported from FlowJo.

```r
s <-  3 # sample number
data <- exprs(gs@data[[s]])
head(data)
```

```
        FSC-A  FSC-H    FSC-W    SSC-A SSC-H    SSC-W Comp-PE-A
[1,] 169366.58 128283 86524.39 63138.69 54779 75537.30  29.93590
[2,] 145468.56 100582 94782.64 82188.27 69586 77404.80  19.39655
[3,] 135920.16 100470 88659.93 53039.61 46870 74162.66  44.84264
[4,] 121107.00  82767 95894.12 84879.90 71531 77766.14  59.92522
[5,] 124500.77  97207 83937.19 43174.62 38739 73039.88  36.18717
[6,]  11673.43  11070 69108.39  5941.35  5802 67110.02  50.22970
     Comp-APC-A Comp-FITC-A Comp-640-A-A Comp-BUV 395-A Comp-BV510-A  Time
[1,]  129.87708    51.53669     42.42691      40.446030    180.03268 0.613
[2,]  127.46664    54.71684     39.96397       8.473352    190.05092 0.615
[3,]  133.58005    53.47903     50.55981      56.433731    172.99860 0.615
[4,]  141.69080    65.89689     50.58980      38.403183    185.50795 0.615
[5,]   39.93256    50.88087     43.56737     131.534653     55.13123 0.617
[6,]   33.90350    45.97820     37.56026      32.189224     69.36352 0.617
```

The markers used to characterize the lymphocytes are the ones included in columns 7 to 11.

```r
data <- data[,7:11]
data <- as.data.frame(data)
```

The data frame can be interrogated for each of the nodes represented in Figure 10 producing a logical list indicating whether the examples are included or not in the gate defined at the node. This functionality has been used to label the examples.

Getting the nodes:

```r
(nodes <- getNodes(gs, path = "auto"))
```

```
 [1] "root"
 [2] "Lymphocytes"
 [3] "Single Cells"
 [4] "Single Cells SSC"
 [5] "Time, NEG viab-CD14 subset"
 [6] "CD56+ CD3+ NKT cell subset"
 [7] "CD56+ CD3+ NKT cell subset/Q1: CD8a- , CD4+"
 [8] "CD56+ CD3+ NKT cell subset/Q2: CD8a+ , CD4+"
 [9] "CD56+ CD3+ NKT cell subset/Q3: CD8a+ , CD4-"
[10] "CD56+ CD3+ NKT cell subset/Q4: CD8a- , CD4-"
[11] "CD56+ CD3- NK cell subset"
[12] "CD56+ CD3- NK cell subset/Q1: CD8a- , CD4+"
[13] "CD56+ CD3- NK cell subset/Q2: CD8a+ , CD4+"
[14] "CD56+ CD3- NK cell subset/Q3: CD8a+ , CD4-"
```

```
[15] "CD56+ CD3- NK cell subset/Q4: CD8a- , CD4-"
[16] "CD56- CD3+ T cell subset"
[17] "CD56- CD3+ T cell subset/Q1: CD8a- , CD4+"
[18] "CD56- CD3+ T cell subset/Q2: CD8a+ , CD4+"
[19] "CD56- CD3+ T cell subset/Q3: CD8a+ , CD4-"
[20] "CD56- CD3+ T cell subset/Q4: CD8a- , CD4-"
[21] "CD56- CD3- non T non NK cell subset"
[22] "B cell subset"
[23] "B cell subset/Q1: CD8a- , CD4+"
[24] "B cell subset/Q2: CD8a+ , CD4+"
[25] "B cell subset/Q3: CD8a+ , CD4-"
[26] "B cell subset/Q4: CD8a- , CD4-"
[27] "CD56-, CD3-, CD19-"
[28] "CD56-, CD3-, CD19-/Q1: CD8a- , CD4+"
[29] "CD56-, CD3-, CD19-/Q2: CD8a+ , CD4+"
[30] "CD56-, CD3-, CD19-/Q3: CD8a+ , CD4-"
[31] "CD56-, CD3-, CD19-/Q4: CD8a- , CD4-"
```

5.1.1.2 Labelling

Cells
Labelling the viable, single cells lymphocytes (no monocytes): node 5.

```
nodes[5]
```

```
[1] "Time, NEG viab-CD14 subset"
```

```
cells <- getIndices(gs[[s]], nodes[5])
table(cells)
```

```
cells
 FALSE   TRUE
295426 139230
```

```
data$cells <-  "other_events"
data$cells <- ifelse(cells, "lymphocytes", data$cells)
data$cells <- factor(data$cells)
table(data$cells)
```

```

 lymphocytes other_events
      139230       295426
```

Subpopulations
Afterwards, the 20 potential subpopulations have been labelled. All the ungated cells will be tagged as "outliers".

Naming the labels:

```
label_names <-
  c("7" = "NKT_4", "8" = "NKT_DP", "9" = "NKT_8", "10" = "NKT_DN",
    "12" = "NK_4", "13" = "NK_DP", "14" = "NK_8", "15" = "NK_DN",
    "17" = "T_cells_4", "18" = "T_cells_DP",
    "19" = "T_cells_8", "20" = "T_cells_DN",
    "23" = "B_4", "24" = "B_DP", "25" = "B_8", "26" = "B_DN",
    "28" = "NO_BTNK_4", "29" = "NO_BTNK_DP",
    "30" = "NO_BTNK_8", "31" = "NO_BTNK_DN")
names(label_names)
```

```
 [1] "7"  "8"  "9"  "10" "12" "13" "14" "15" "17" "18" "19" "20" "23" "24"
[15] "25" "26" "28" "29" "30" "31"
```

Labelling the cells:

```
for(i in names(label_names)){
    indices <- getIndices(gs[[s]], nodes[as.numeric(i)])
    # if index for node i is TRUE, data$label takes label_name[i]
    # otherwise, it remains unchanged
    data$labels <- ifelse(indices, label_names[[i]], data$labels)
}

data$labels <- as.factor(data$labels)
table(data$labels)
```

```

       B_4        B_8       B_DN       NK_4       NK_8      NK_DN
        69         72      14369        458      14615      12314
     NK_DP      NKT_4      NKT_8     NKT_DN     NKT_DP  NO_BTNK_4
       474        569       3023        979        218       1328
 NO_BTNK_8 NO_BTNK_DN NO_BTNK_DP   outliers  T_cells_4  T_cells_8
       446       4636        107     295762      62113      17625
T_cells_DN T_cells_DP
      4209       1270
```

Main populations

The populations of interest are easily extracted using 2 genreic lists: `metalabels_list` and `patterns_list`:

```
# Empty column
data$metalabels <- "outliers"

# Naming the meta-labels and the patterns that will be used to define them
metalabels_list <- c("NKT", "NK", "T", "B", "NO_BTNK")
patterns_list <- c("NKT_", "NK_", "T_cells_", "B_", "NO_B")
```

```r
for(i in 1:length(metalabels_list)){
  data$metalabels <-
    ifelse(grepl(patterns_list[i], data$labels),
           metalabels_list[i], data$metalabels)
}

data$metalabels <- factor(data$metalabels,
                          levels = c(metalabels_list, "outliers"))
table(data$metalabels)
```

```
     NKT        NK         T         B   NO_BTNK  outliers 
    4789     27861     85217     14510      6517    295762 
```

Lymphocytes
Finally, all the events not included in the gate delimiting the single-cell, living lymphocytes are excluded:

```r
data <- data[data$cells == "lymphocytes",]
table(data$labels, data$metalabels)
```

```
              
                 NKT    NK     T     B NO_BTNK outliers
  B_4              0     0     0    69       0        0
  B_8              0     0     0    72       0        0
  B_DN             0     0     0 14369       0        0
  NK_4             0   458     0     0       0        0
  NK_8             0 14615     0     0       0        0
  NK_DN            0 12314     0     0       0        0
  NK_DP            0   474     0     0       0        0
  NKT_4          569     0     0     0       0        0
  NKT_8         3023     0     0     0       0        0
  NKT_DN         979     0     0     0       0        0
  NKT_DP         218     0     0     0       0        0
  NO_BTNK_4        0     0     0     0    1328        0
  NO_BTNK_8        0     0     0     0     446        0
  NO_BTNK_DN       0     0     0     0    4636        0
  NO_BTNK_DP       0     0     0     0     107        0
  outliers         0     0     0     0       0      336
  T_cells_4        0     0 62113     0       0        0
  T_cells_8        0     0 17625     0       0        0
  T_cells_DN       0     0  4209     0       0        0
  T_cells_DP       0     0  1270     0       0        0
```

5.1.1.3 Down-sampling

The `createDataPartition()` function from the `caret` package is used. This function generates partitions with representative frequencies for the indicated classes. The partitioning is based on the tag "labels" (the subpopulations) according the number of cells indicated in the parameters:

```r
p <- params$cells/nrow(data)
set.seed(42)
subsample <- createDataPartition(data$labels, p = p, list = F)
data <- data[subsample, ]
```

The data is finally exported as an `.fcs` file. Label lists are exported as `RDS` objects.

5.1.2 Algorithm benchmarking script

5.1.2.1 Number of clusters

`FlowSOM` clustering is performed by specifying the number of clusters. For the other algorithms it cannot be specified. The 5-parameter experimental design uses 5 markers to define 5 main cell populations, 4 of them with biological interest. One of the main populations (T cells) can be divided in 4 subpopulations of interest. According to these premises, `FlowSOM` has been conducted with the following number of clusters:

- 4: The number of populations a researcher may expect to find (NKT, NK, T and B cells)
- 5: Main populations (NKT, NK, T, B cells and other lymphocytes)
- 8: Main populations + T subpopulations (NKT, NK, T4, T8, T double positive, T double negative, B cells and other lymphocytes)
- 10, 20: A higher number of clusters may help finding rare populations. 20 is the number of potential subpopulations.

5.1.2.2 Functions

The function for the matching procedure is the same that has been used for the synthetic samples. On the contrary, the mean F1 is now computed taking into consideration exclusively the populations that have been predicted:

```r
mean_f1 <- function(cm){
  # Extracting the F1 values
  # cm: confusion matrix element
  f1_list <- cm$byClass[,"F1"]
  # removing NAs
  f1_list <- f1_list[!is.na(f1_list)]
  # Computing mean F1
  return(mean(f1_list))
}
```

5.1.2.3 Clustering

Runtimes are measured for each clustering algorithm using the `proc.time()` function. For example:

```
# CLUSTERING
ptm <- proc.time()
pred <- Rphenograph(ff@exprs)
clustering <- pred$membership
ptm <- proc.time() - ptm
```

After the clustering, a first round of matching is performed against the main populations. Afterwards, a second round of matching is run with the populations that have been split into several clusters, if that is the case:

```
#### SPLIT POPULATIONS
  # How many clusters have been assigned to each population?
  table_maxs <- table(matched$m)
  # Which populations have been assigned to more than one cluster?
  split_list <-  names(table_maxs[table_maxs>1])
  split_list <- as.numeric(split_list)

  # If there are, start MATCHING SUB-LABELS AND PREDICTORS,
  # one population at a time
  if(length(split_list) != 0){
    for(j in 1:length(split_list)){
    # counter
    k <-  k + 1
    # Name of the population
    name_pop <- levels(metalabels)[split_list[j]]
    # Selecting the well-predicted cells
    # (e.g., all the T cells matched as T)
    well_pred <- rep(NA, length(metalabels))
    well_pred <- ifelse(main_matched == name_pop &
                          metalabels == name_pop, TRUE, FALSE)

    # MATCHING SUBLABELS AND PREDICTIONS
    # Only the clusters for the population
    table_clusters <- table(clustering[well_pred])
    clusters_list <- names(table_clusters[table_clusters>0])
    # Only the sublabels for the population
    # Only the levels for those sublabels
    sublevels <- sublabels_levels[[name_pop]]
    matched <- matching(
      factor(clustering[well_pred], levels = clusters_list),
      factor(sublabels[well_pred], levels = sublevels))

    # COMPUTING THE CONFUSION MATRIX AND OTHER PERFORMANCE MEASUREMENTS
    cm <- confusionMatrix(data = matched$preds,
                          reference = factor(
                            sublabels[well_pred], levels = sublevels))
    # Add the F1 values to a list
    subs_f1 <- c(subs_f1, cm$byClass[, "F1"])

    # COMPUTE THE MEAN F1
    mf1 <- mean_f1(cm)
    # Storing the result
    summ_subs[k,] <- c(
      params$file, nrow(ff@exprs), method, nc, number_partitions,
      name_pop, f1_temp[[split_list[j]]], table_maxs[table_maxs>1][[j]],
      length(table(matched$m)), mf1)
```

The procedure has been explained in detail in the eighth deliverable, available upon request. The whole benchmarking script can be found at the GitHub repository.

5.1.3 Results

5.1.3.1 Samples

Three samples from different blood donors have been analyzed. The frequencies for the main populations and subpopulations obtained with the manual gating strategy are shown in Figure 11. Samples 2 and 3 have been down-sampled to 25,000, 50,000 and 100,000 examples. Sample 1 had fewer than 100,000 examples (single-cell, living lymphocytes), thus, only the 25,000 and 50,000 down-samples have been performed. Finally, the method UMAP + `ClusterX` could not be applied to the Sample 3, 1000,000 cells, because `ClusterX` requires too much computing power compared to other methods.

5.1.3.2 FlowSOM: selection of the better condition

The results obtained with the number of clusters tested on `FlowSOM` are compared in terms of mean F1 score, number of matched partitions (detected populations) and CPU runtimes. Figure 12 shows the results obtained for the different down-samples. In some cases, the lines might overlap.

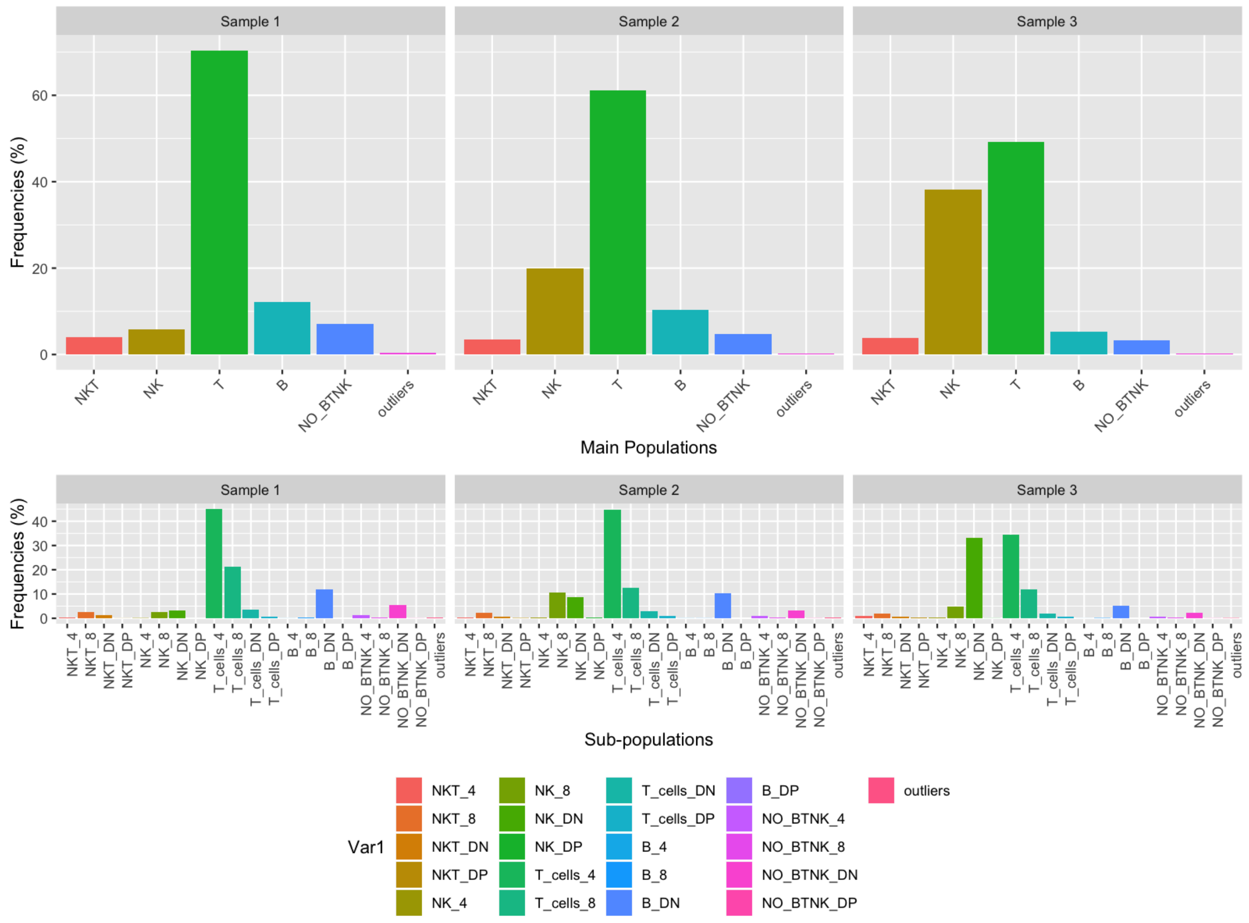


Figure 11: Cell frequencies for the 3 samples analyzed for the 5-parameter design, obtained by manual gating. Frequencies for the 5 main populations and the potential 20 subpopulations are indicated, as well as the outliers.

The best mean F1 scores are obtained at 20 clusters for all the samples. Below 20 clusters, the mean F1 scores are highly variable among samples. This number of clusters allows finding the 5 partitions in all the conditions tested. Increasing the number of clusters clearly increases the runtime, but it remains fast anyway. The condition with 20 clusters is thus selected to be compared to the other algorithms tested.

5.1.3.3 Mean F1

The mean F1 scores obtained with `FlowSOM` (20 clusters), `RPhenograph` , UMAP + `flowPeaks` and UMAP + `ClusterX` are compared in terms of number of found clusters and number of cells (Figure 13). Please notice that the scale for the mean F1 is narrower in this figure than in the previous one.

The number of cells does not seem to influence the mean F1 scores. The UMAP + `flowPeaks` method achieves similar mean F1 scores than the other methods but with fewer clusters, getting closer to the number of main populations (5). The other methods are closer to the number of potential subpopulations (20).

5.1.3.4 Number of matched partitions

The number of matched partitions (detected populations) is compared to the number of detected clusters (Figure 14). `FlowSOM` (20 clusters), `RPhenograph` and UMAP + `ClusterX` methods find the five populations in all the conditions tested. Nevertheless, the number of clusters found by these methods is high. On the contrary, UMAP + `flowPeaks` manages to detect all the populations with fewer number of clusters. On the contrary, it seems to need a higher number of cells, as for 2 of the 25,000 down-samples only 4 populations are detected.

5.1.3.5 CPU runtime

Runtimes are recorded for all methods and samples combinations (Figure 15). The runtime for UMAP is added to the runtimes for `flowPeaks` or `ClusterX`. Both `FlowSOM` and `RPhenograph` are computationally efficient in the conditions tested. The runtime used by UMAP + `flowPeaks` increases noticeably as the number of cells and the number of detected clusters increases but remains reasonable in comparison with the runtimes achieved with the UMAP + `CLusterX` method, extremely sensitive to the number of cells. In fact, the clustering with one of the 100,000 samples could not been achieved with this latter method due to insufficient computing power. For this reason, the UMAP + `ClusterX` method has been discarded.

5.1.3.6 Analysis of split populations

The populations that have been matched to more than one cluster have been compared to their corresponding subpopulations. For example, if the population T has been detected by more than one cluster, all the clusters matching to this population have been compared to the T subpopulations: T4, T8, T_DP and T_DN. The matching procedure is applied, the

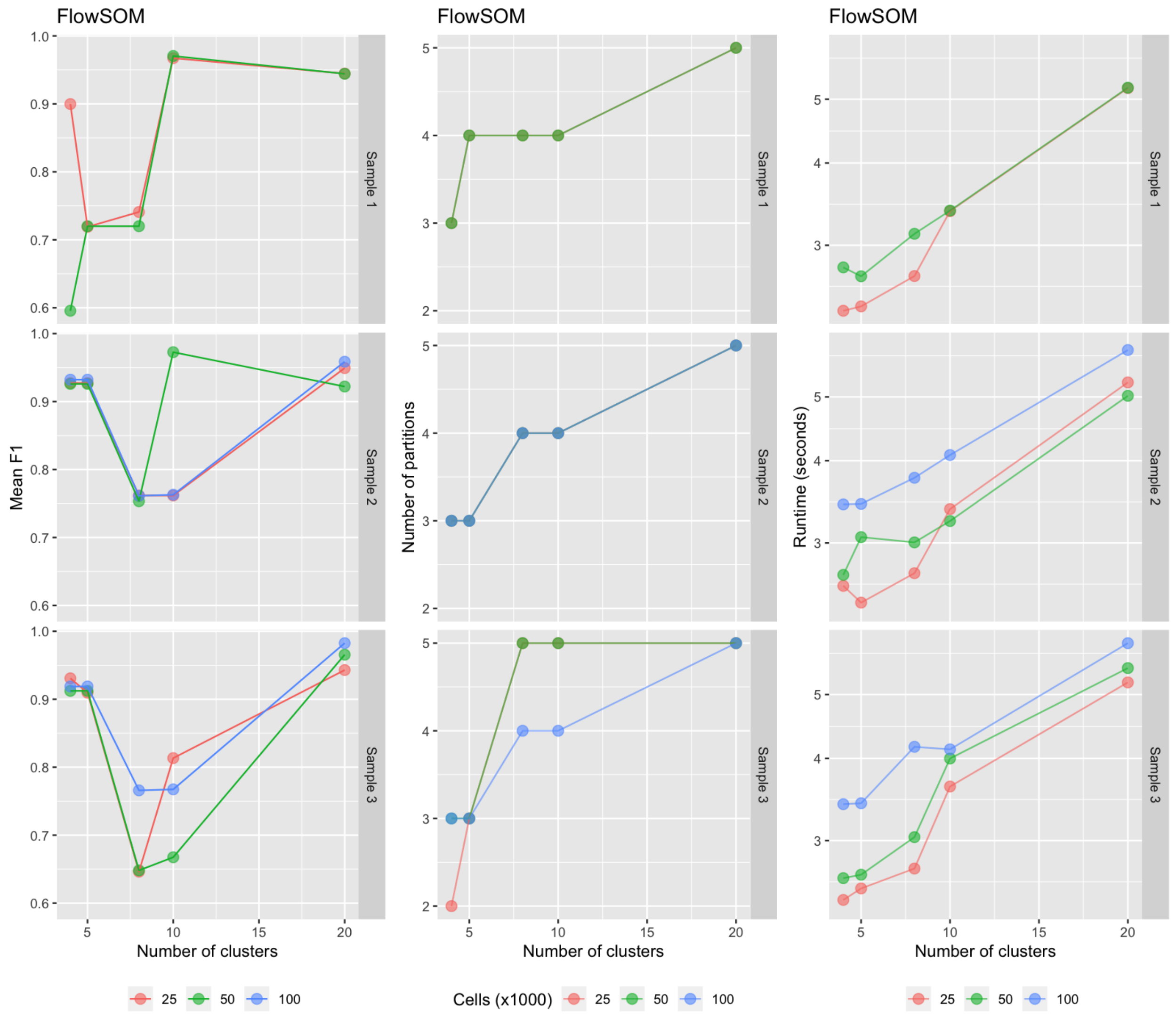


Figure 12: 5-parameter design. FlowSOM clustering has been performed specifying to find 4, 5, 8, 10 or 20 clusters. Sample 1, n=2 down-samples; samples 2 and 3, n=3 down-samples.

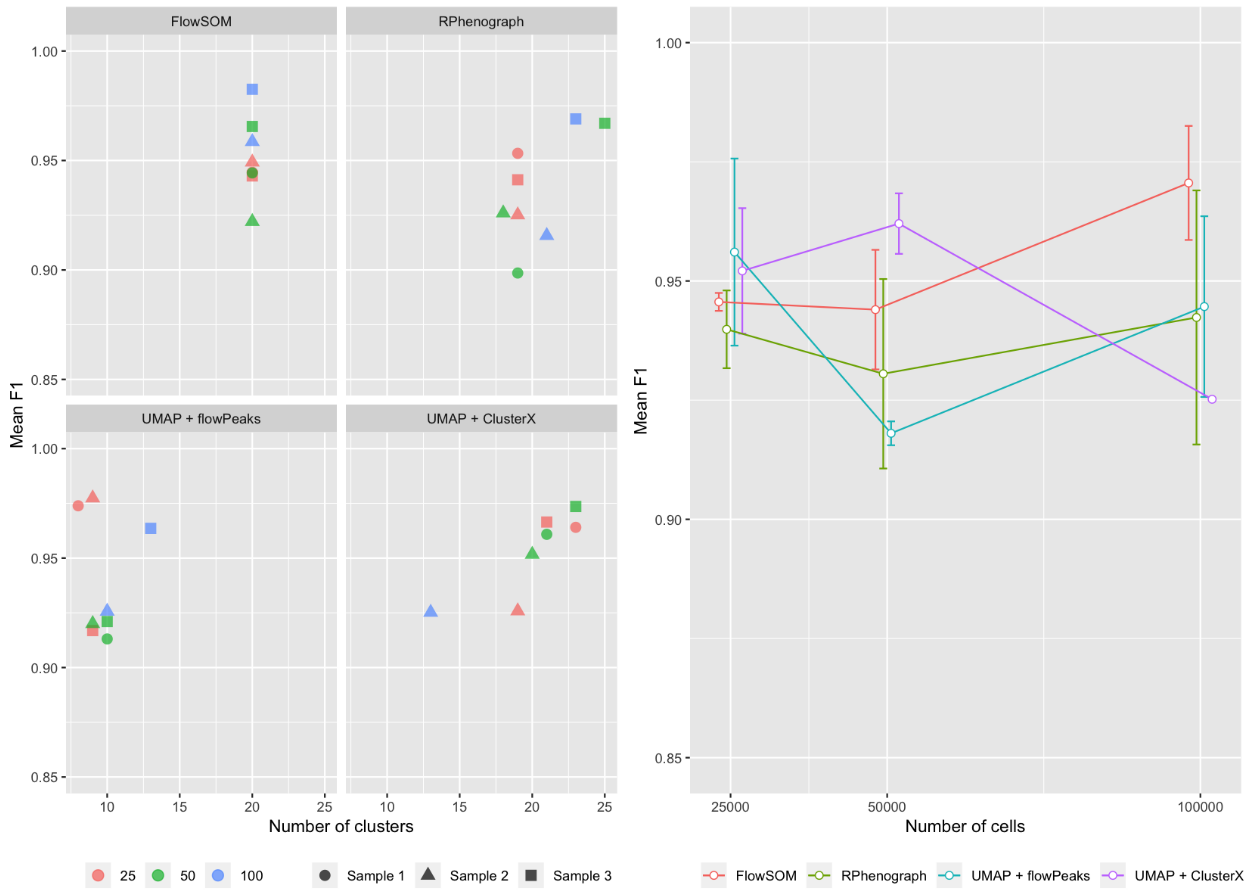


Figure 13: 5-parameter design. Mean F1 scores related to the number of predicted clusters (left panel) and the number of cells (right panel, mean ± standard error). Sample1, n=2; samples 2 and 3, n=3. FlowSOM, RPhenograph and UMAP + flowPeaks, 25,000 and 50,000 cells, n=3; 100,000 cells, n=2. UMAP + ClusterX 25,000 and 50,000 cells, n=2; 100,000 cells, n=1.

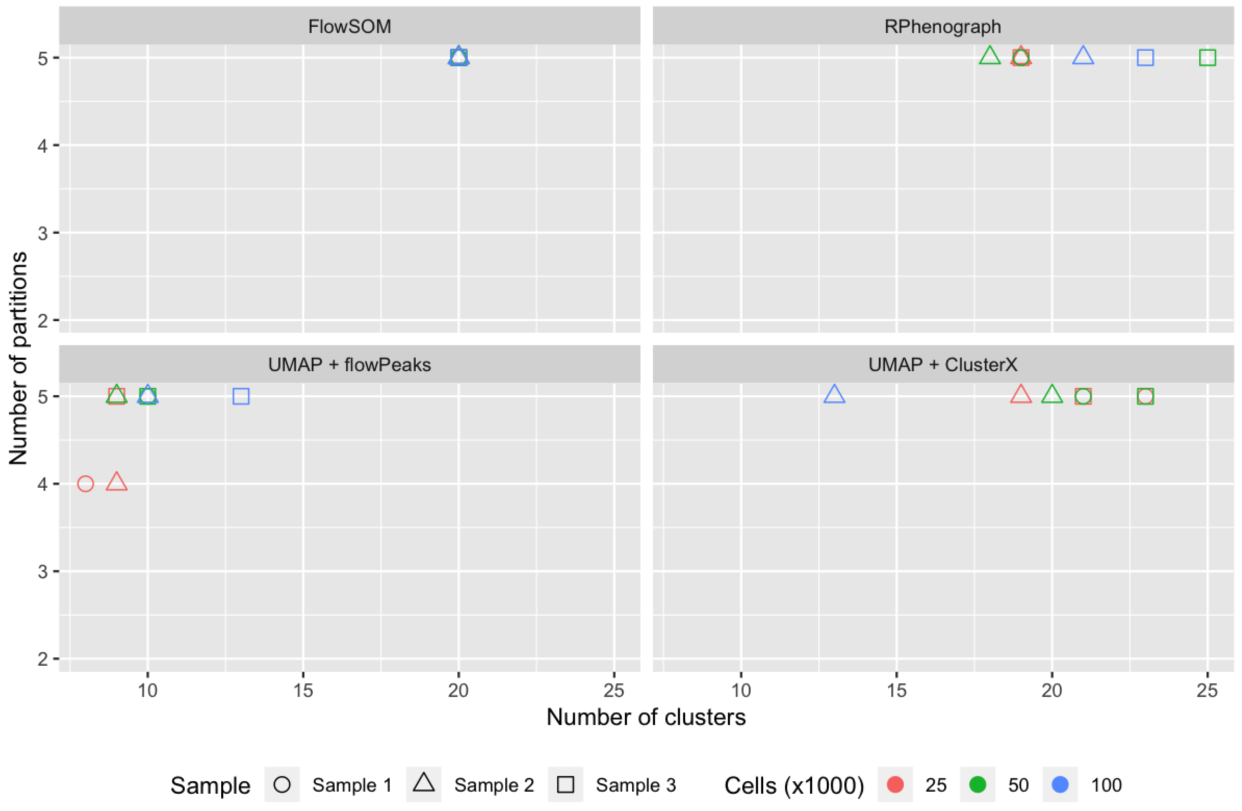


Figure 14: 5-parameters design. Number of matched partitions (detected populations) related to the number of clusters. Sample 1, n=2; samples 2 and 3, n=3. FlowSOM, RPhenograph and UMAP + flowPeaks, 25,000 and 50,000 cells, n=3; 100,000 cells, n=2. UMAP + ClusterX 25,000 and 50,000 cells, n=2; 100,000 cells, n=1.

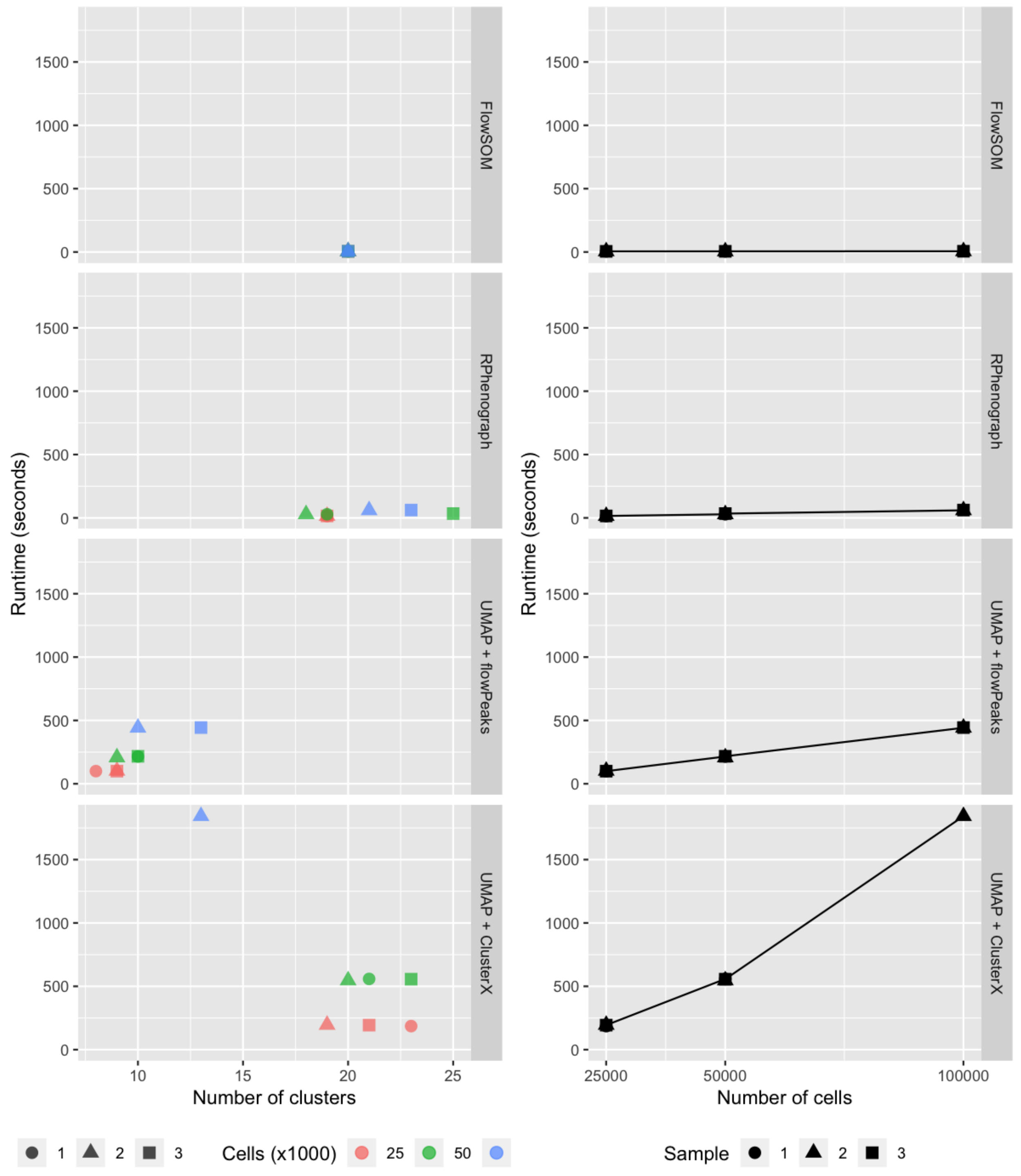


Figure 15: 5-paramenter design. CPU (user) runtimes related to the number of detected clusters (left panel) and the number of cells (right panel). Sample 1, n=2; samples 2 and 3, n=3. FlowSOM, RPhenograph and UMAP + flowPeaks, 25,000 and 50,000 cells, n=3; 100,000 cells, n=2. UMAP + ClusterX 25,000 and 50,000 cells, n=2; 100,000 cells, n=1.

F1 scores computed for each of the subpopulations and averaged for the subpopulations that have been matched, obtaining a final mean F1 score. The objective of this analysis is to determine if the algorithms split the populations according its phenotypic profile or in an apparently random manner.

Results obtained for the samples containing 25,000 and 50,000 cells are shown in Figure 16. On the first vertical panels for each down-sample condition (1st and 3th), the F1 scores for the matched partitions are indicated. On the right panels (2nd and 4th), the F1 scores for the populations that had been matched to more than one cluster are related to the mean F1 scores obtained with the analysis of the subpopulations.

When the samples are close to reach the point (1,1), optimal F1 and mean F1 scores are obtained: when populations are split, they are correctly matched to the reference subpopulations. This is the case for all the methods except for `FlowSOM`. If the samples appear above the diagonal, as it happens with some NKT cells with `FlowSOM`, it indicates good F1 scores but poor accuracy for the subpopulations. On the contrary, samples located below the diagonal indicate that populations that had bad F1 scores have been correctly split. Interestingly, `RPhenograph` achieves good mean F1 scores for subpopulations of NKT having poor F1 scores. Furthermore, the small size of these samples on the plot indicates that these cells have been matched to a small number of clusters, probably 1 or 2. This could explain the poor F1 score: only a part of the NKT population has been detected. The analysis of the subpopulations allows to determine that, nevertheless, the part that has been detected matches correctly with referenced subpopulations. In this sense, `RPhenograph` appears to be a powerful algorithm to detect small populations.

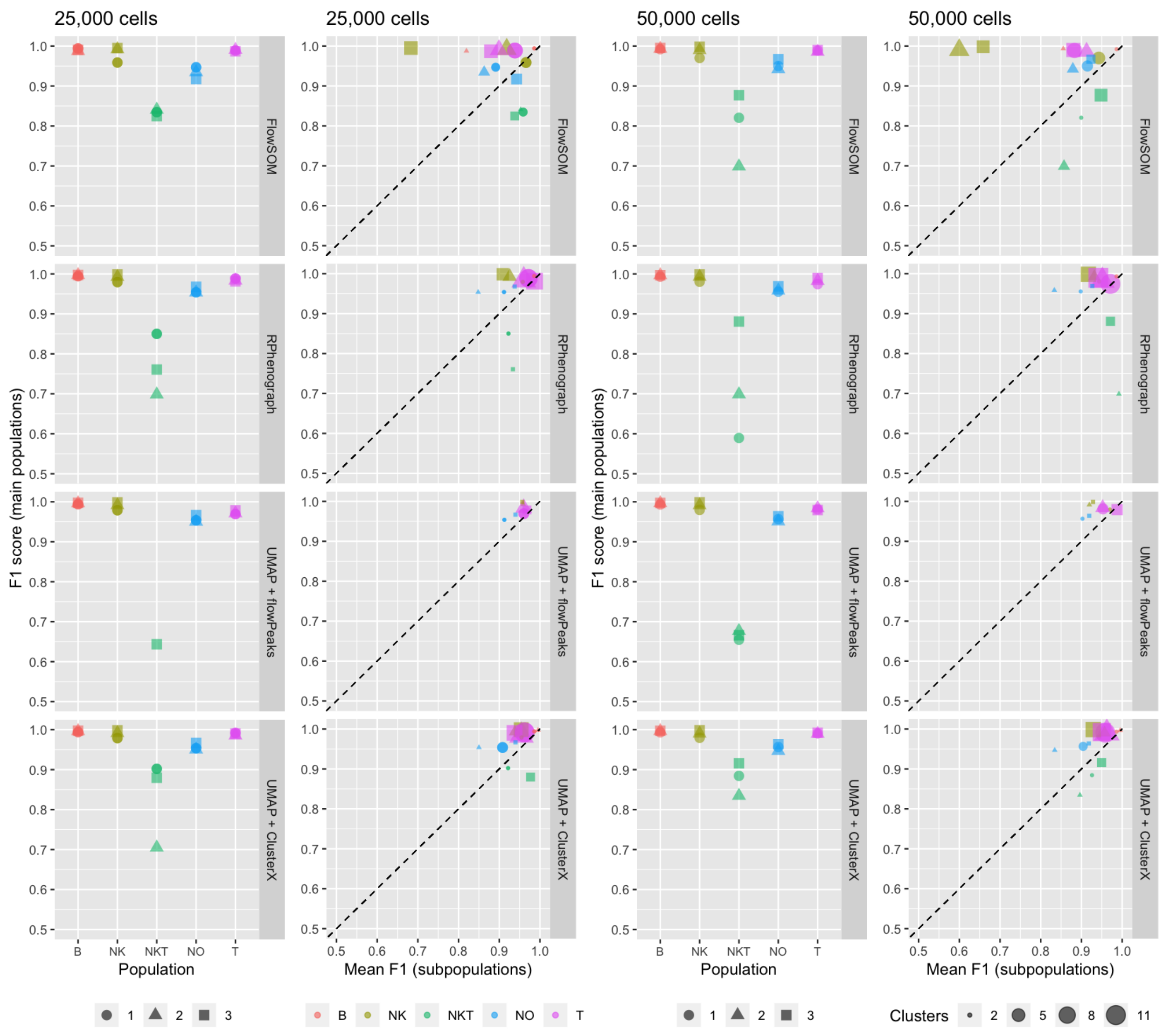


Figure 16: 5-parameter design. Anlysis of subpopulations on split main populations. 1st and 2nd panels: 25,000 cell down-samples. 3th and 4th panels: 50,000 cell down-samples. 1st and 3th panels: The F1 scores are indicated for all the matched partitions. 2nd and 4th panels: The F1 scores obtained for the matched partitions corresponding to the main populations are related to the mean F1 scores computed for the populations that have been split in more than one cluster. Dot, square and triangle sizes are drown proportional to the number of clusters found for each population. Notice that NO stands for NO_BTNK cells.

5.2 11-parameter design

Data is manually gated in `FlowJO` as depicted in Figure 17. Lymphocytes are divided onto 19 populations with biological significance.

5.2.1 Data preparation

An `R Markdown` script has been developed in order to prepare the 11-parameter flow cytometry data for algorithm benchmarking. The whole script is available at the GitHub repository. The script is a simplified version of the preparing script used for the 5-parameter experimental design. In fact, the complexity of expression patterns detected with 11 markers does not allow to perform an analysis of subpopulations with just two levels of complexity as it has been done with the 5-parameter design. Thus, the clustering results have been compared to a unique label list containing 19 populations of biological interest. The gating strategy imported from `FlowJo` is shown in Figure 18.

5.2.2 Algorithm benchmarking script

The script used for the 11-parameter experimental design is a simplified version of the one used for the 5-parameter design. There is no second matching round on subpopulations, and the UMAP + `ClusterX` method has been eliminated. The script is available at the GitHub repository.

5.2.3 Results

5.2.3.1 Samples

Two samples from different blood donors have been analyzed. Frequencies for the 19 populations obtained with the manual gating strategy are shown in Figure 19. The samples have been down-sampled to 100,000, 200,000, 300,000 and 400,000 examples.

5.2.3.2 FlowSOM: selection of the better condition

`FlowSOM` clustering has been performed for 5, 10, 15, 20, 30, 40, 50 and 60 clusters. Figure 20 shows the results obtained in terms of mean F1 score, number of matched partitions (detected populations) and CPU runtimes. In all the cases, the running times are excellent.

Whereas the number of clusters does not have a clear effect on the mean F1 score, it does influence the number of matched partitions, getting closer to the 19 populations of reference as the number of clusters increases. From 40 to 60 clusters, sample 1 experiences a slight improvement in the mean F1 score, but the number of partitions remains quite stable; there is no clear amelioration for sample 2 neither. The slight improvements that are observed for some of the down-samples with 60 clusters appear to be random and, in any case, insufficient

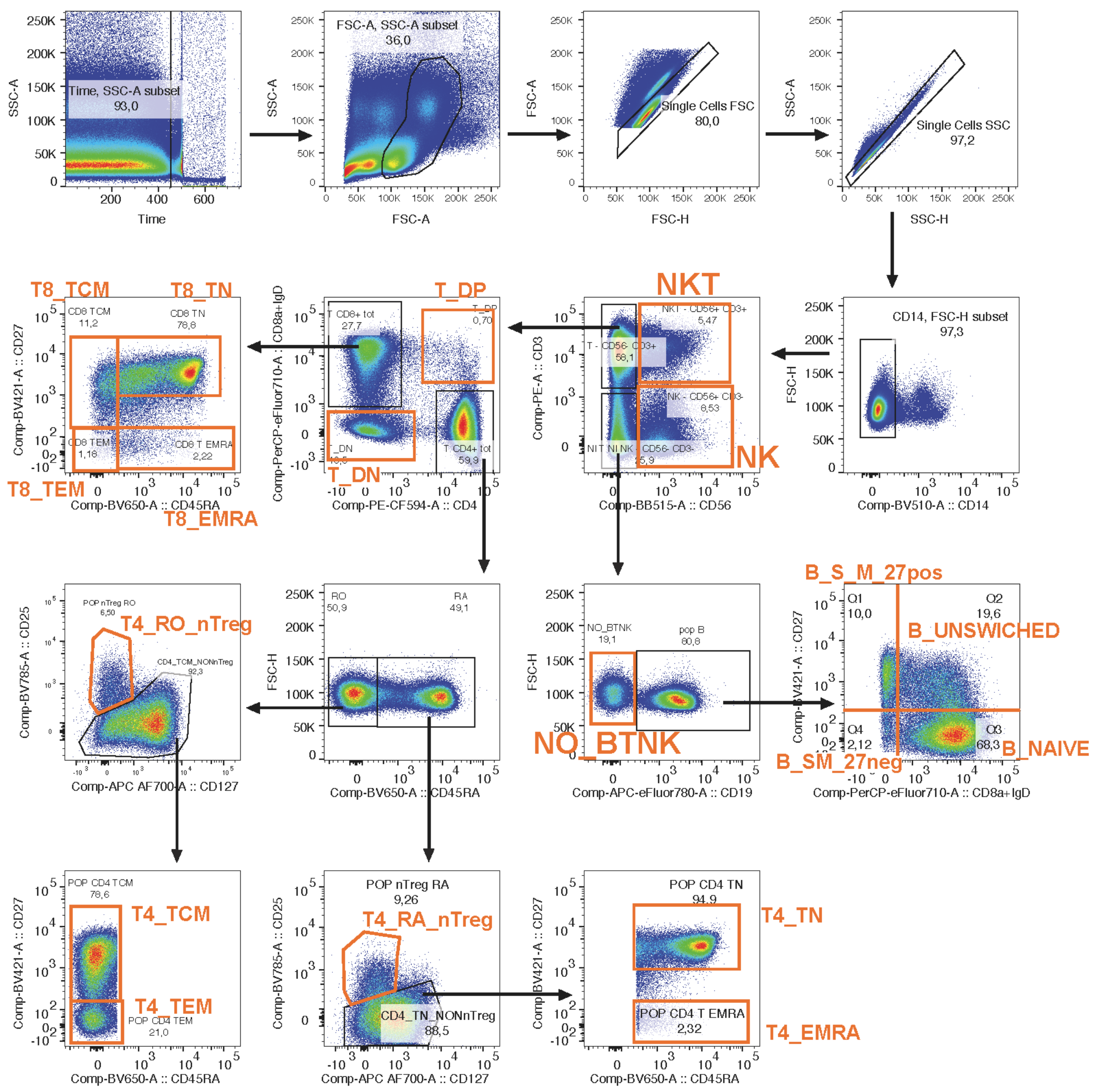


Figure 17: Manual gating strategy for the 11-parameter experimental design. 19 populations of lymphocytes are delimited (orange).

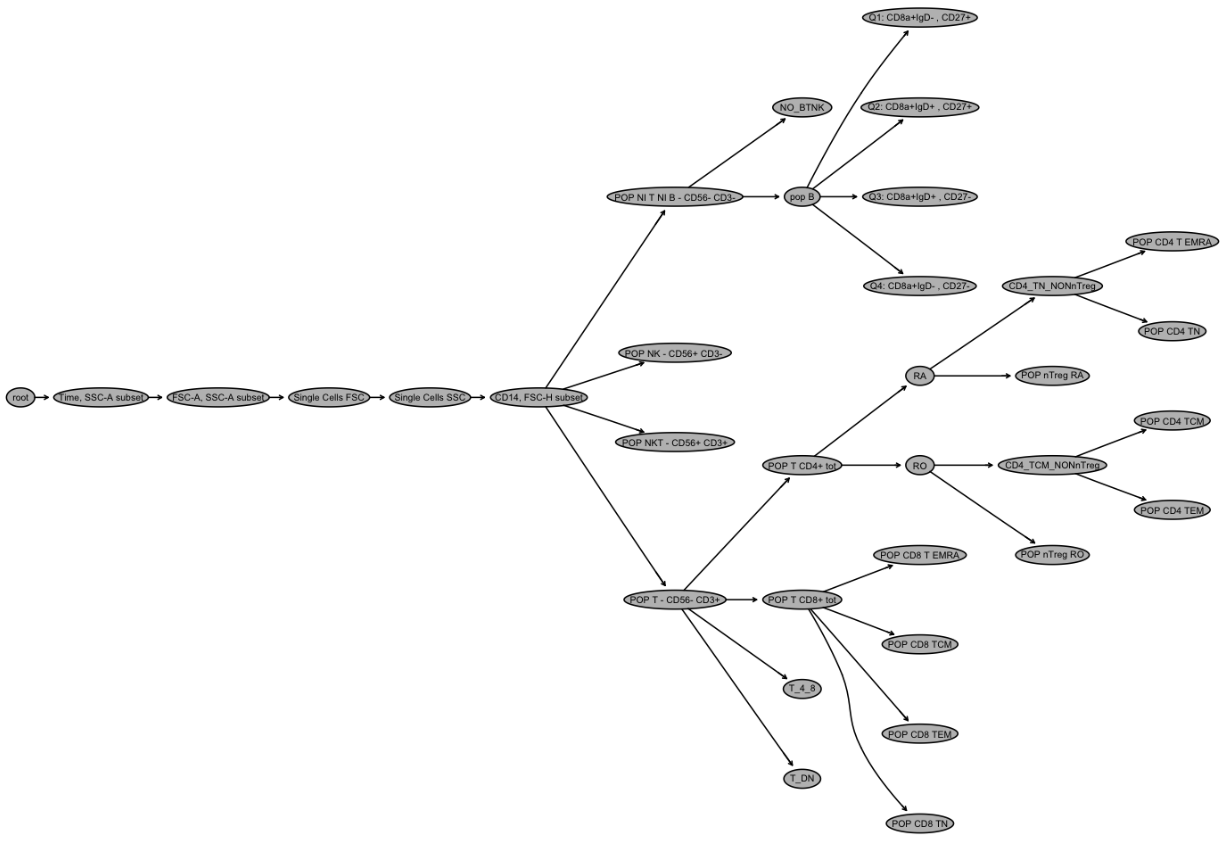


Figure 18: Representation of the hierarchical gating strategy for the 11-parameter design imported from FlowJo.

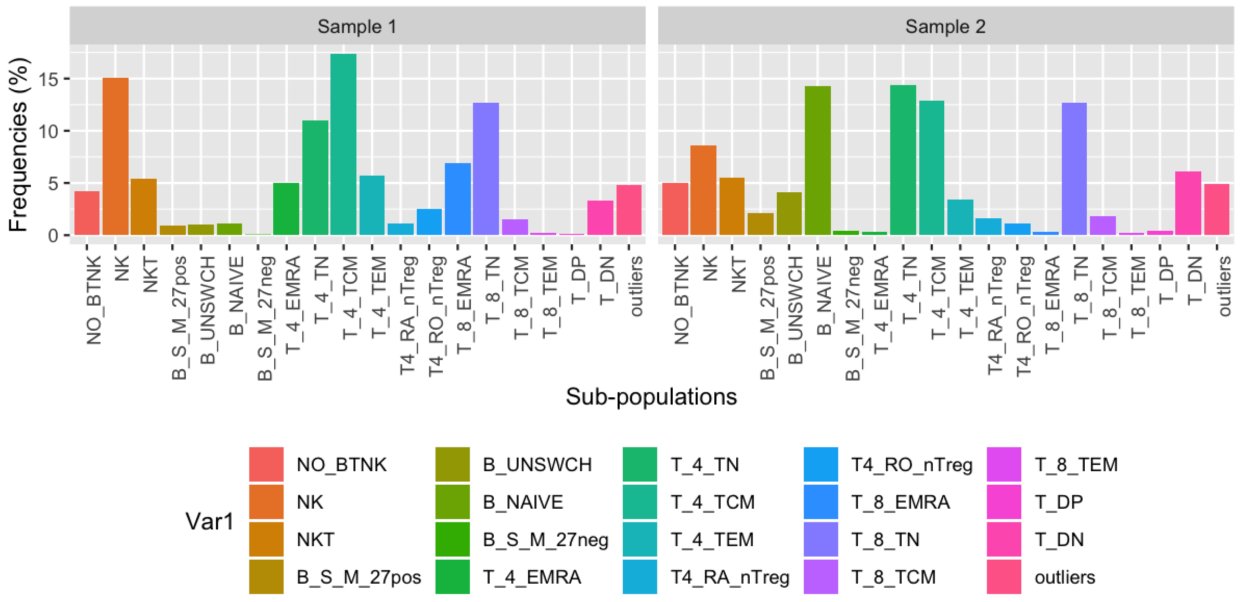


Figure 19: Cell frequencies for the 2 samples analyzed for the 11-parameter design, obtained by manual gating. Frequencies for 19 populations are indicated, as well as the outliers.

to justify using such a number of clusters. Therefore, the condition with 40 clusters is thus selected to be compared to the other tested algorithms.

5.2.3.3 Mean F1

The mean F1 scores obtained with `FlowSOM` (40 clusters), `RPhenograph` and UMAP + `flowPeaks` are compared in terms of number of found clusters and number of cells (Figure 21). None of the methods seems to be affected by the number of cells. However, `RPhenograph` seems to be sensitive to the sample: it reaches better mean F1 scores with sample 1 and founds fewer clusters with sample 2. UMAP + `flowPeaks` seems to be more robust in terms of mean F1 score. Finally, the scores obtained with `FlowSOM` are clearly poorer.

5.2.3.4 Number of matched partitions

The number of matched partitions (detected populations) is compared to the number of detected clusters (Figure 22). Whereas `FlowSOM` reaches the poorest mean F1 scores (Figure 21), it is the method that gets closer to the number of reference populations (which is 19). The number of clusters does not seem to affect the number of matched partitions in `RPhenograph`, that gets its better result with one of the largest down-samples. On the contrary, UMAP + `flowPeaks` number of matched partitions seems to correlate positively with the number of clusters.

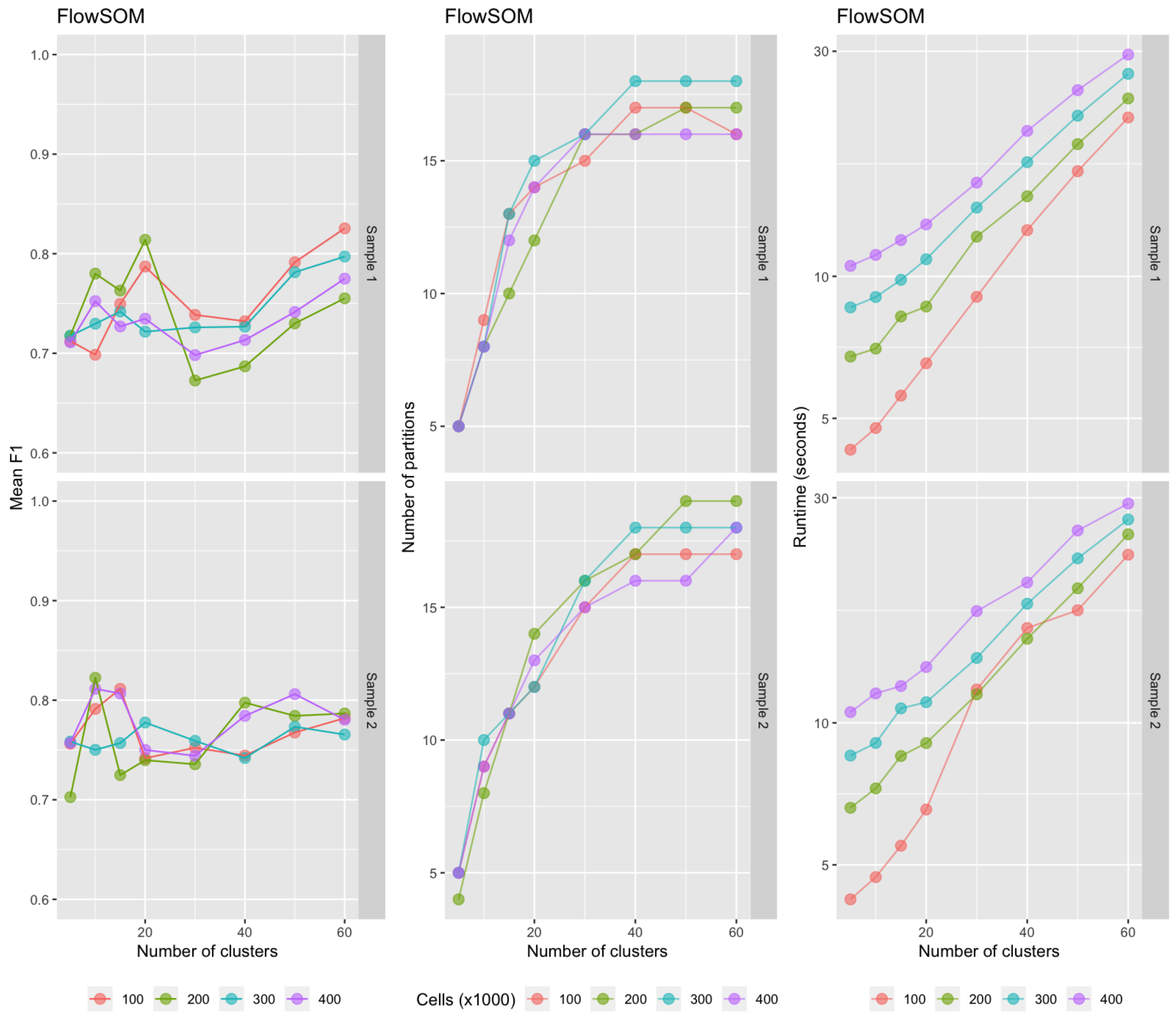


Figure 20: 11-parameter design. FlowSOM clustering has been performed specifying to find 5, 10, 15, 20, 30, 40, 50 or 60 clusters.

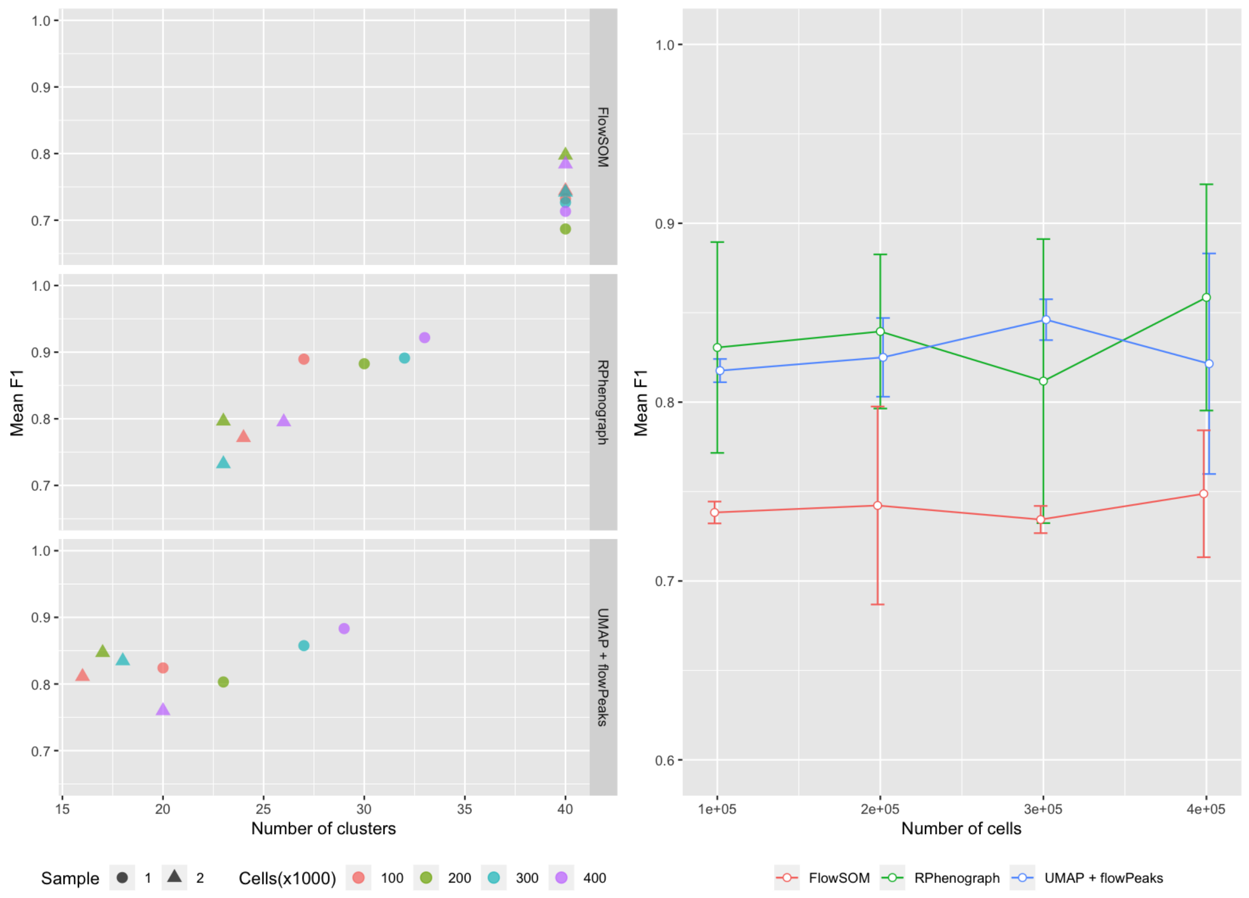


Figure 21: 11-parameter design. Mean F1 scores related to the number of predicted clusters (left panel) and the number of cells (right panel, mean ± standard error). Samples 1 and 2, n=4 down-samples.

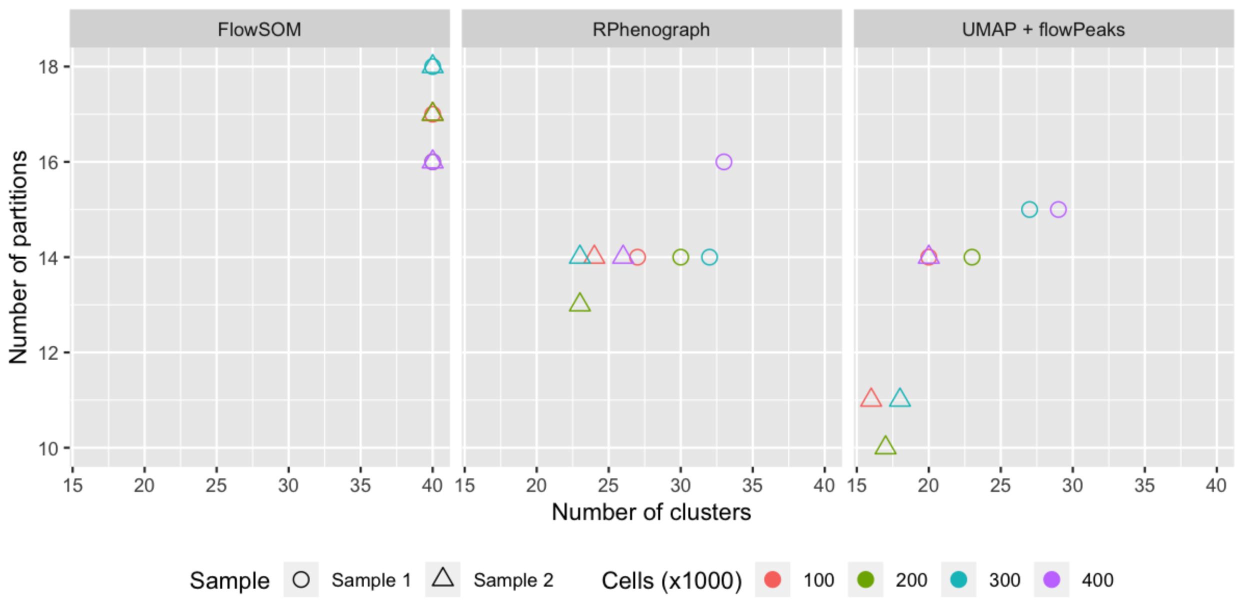


Figure 22: 11-parameters design. Number of matched partitions (detected populations) related to the number of clusters. Samples 1 and 2, n=4 down-samples.

5.2.3.5 CPU runtime

Runtimes are recorded for all methods and samples combinations (Figure 23). The runtime for UMAP is added to the runtimes for `flowPeaks`.

`FlowSOM` is really efficient in terms of computational time in all the conditions tested. `RPhenograph` runtime increases with the cell number; nevertheless, it remains moderate in comparison with the runtimes achieved by the UMAP + `flowPeaks` method, which seems to be sensitive to both cell number and number of clusters.

5.3 Conclusions

Figure 24 summarizes the results obtained with the better conditions tested:

- 5-parameter design : 100,000 cells, FlowSOM number of clusters settled to 20.

- 11-parameter design : 400,000 cells, FlowSOM number of clusters settled to 40.

As it can be observed, increasing the complexity of the experimental design makes more evident the differences on algorithm performances. With 5 parameters, `FlowSOM` achieves equivalent mean f1 scores to the other two methods; on the contrary, it appears to be less performant in the 11-parameter design. Mean F1 scores are poorer and number of clusters is more elevated to those found by other algorithms, making further analyses difficult.

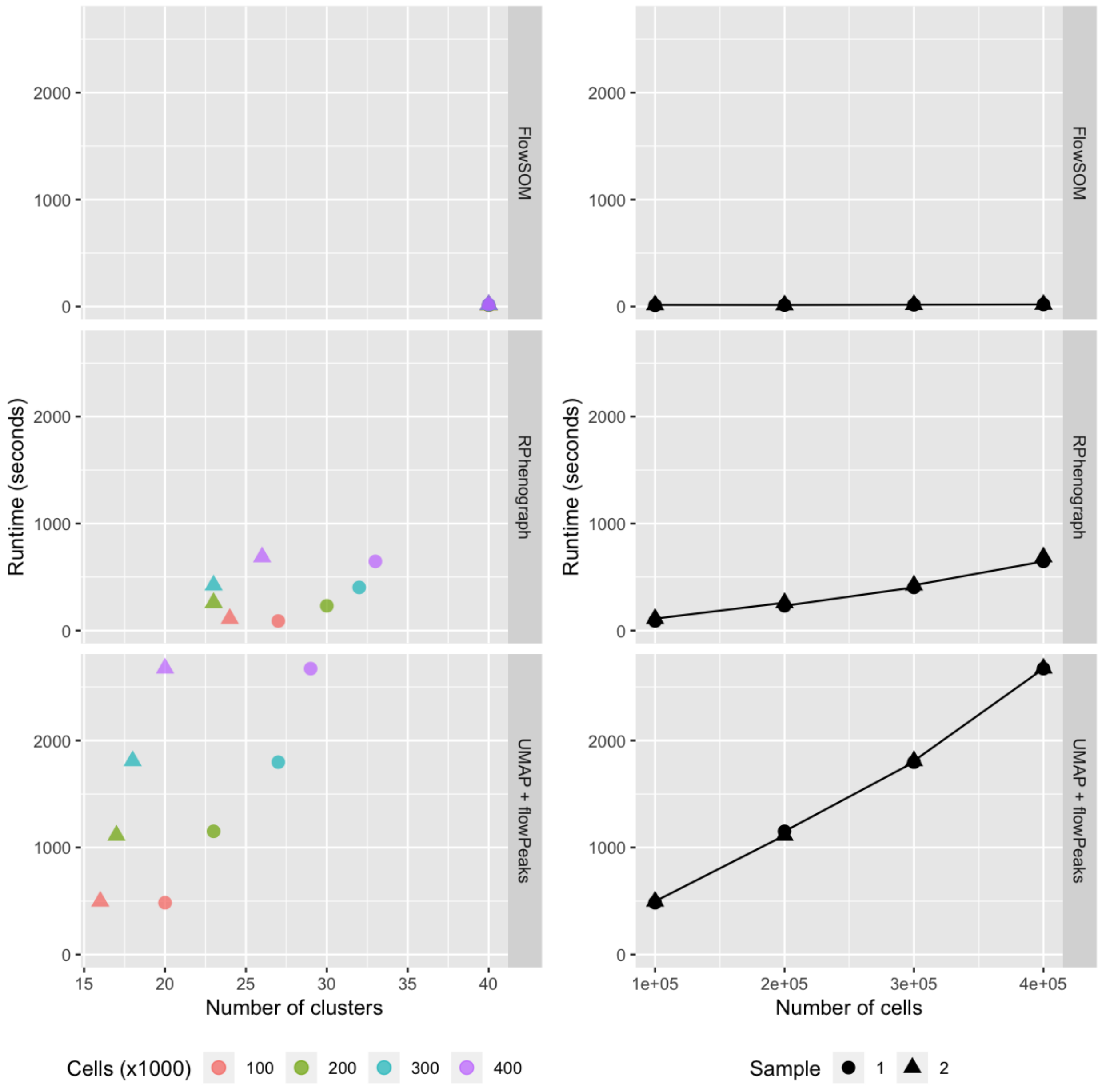


Figure 23: 11-paramenter design. CPU (user) runtimes related to the number of detected clusters (left panel) and the number of cells (right panel).Samples 1 and 2, n=4 down-samples.

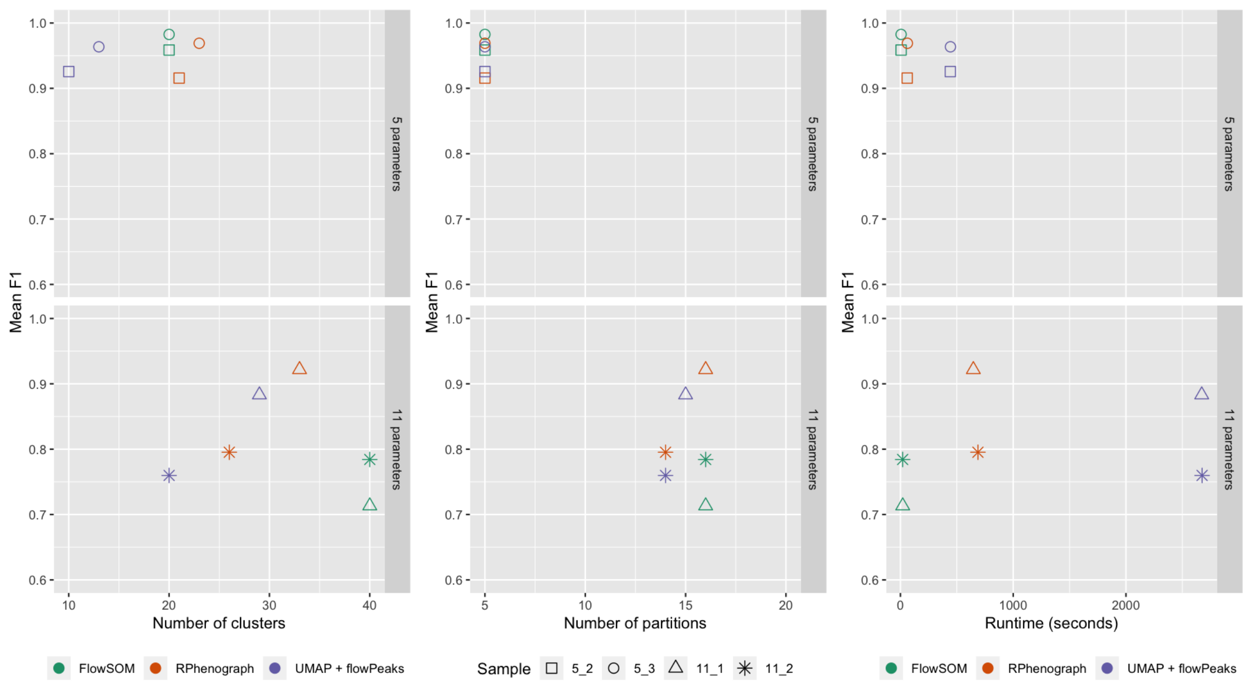


Figure 24: Summary of the results achieved with the methods and conditions tested. 5_2: 5 parameters, sample 2. 5_3: 5 parameters, sample 3. 11_1: 11 parameters, sample 1. 11_2: 11 parameters, sample 2.

The number of matched partitions is also less stable in the 11-parameter design, whereas it remains constant for the 5-parameter examples that are shown in this figure.

Finally, `RPhenohgraph` and UMAP + `flowPeaks` obtain similar mean F1 scores, slightly higher for `RPhenograph` for the 11-parameter samples, and similar numbers of matching partitions. `RPhenograph` seems to need more clusters than UMAP + `flowPeaks` to achieve similar performances, but it is much more competitive in terms of computational time.

To conclude, `RPhenograph` provides the best balance among the different performance measurements that have been tested in this study. Additionally, as it does not need any random start, it appears to be more robust.

6 Discussion

The main objective of this project has been to explore, evaluate and select clustering algorithms for multiparametric data exploration in order to accommodate the selected methods in a graphical front-end. This final goal has been decisive in the choice of software and algorithms, nevertheless, due to the limited time available, this task will be performed afterwards as a logical continuance.

The project was ambitious and not enough developed to properly estimate the duration of the different tasks. Nevertheless, the work that has been performed has allowed to get introduced and familiarized with the methodology and bibliography of the field. Furthermore, the experimental methodology that has been developed and the results that have been obtained will be the core for further analyses at the IRSL Flow Cytometry facility.

First of all, an extensive bibliographic research on unsupervised clustering algorithms applied to cytometry data has been performed. Afterwards, a selection of algorithms has been explored and evaluated by adapting the methodology found in the literature to the specific needs of the project. Furthermore, a script for generating synthetic data has been developed, enabling the performance of tests in strictly controlled conditions. Indeed, the synthetic data has been an excellent material to develop a methodology for performance evaluation. The existing methods used to explore automatic gating algorithms have been adapted to evaluate clustering algorithms as explorative tools. This methodology has then been applied to real flow cytometry data. Two common experimental designs used to characterize human lymphocytes have been used to benchmark a selection of algorithms. In agreement with the literature, the obtained results indicate `RPhenograph` as a powerful clustering algorithm.

In addition to the generated results, several interesting questions have risen during the development of this project. One example: the concept of "population". Cytometrists are used to define cell populations of biological interest based on the differential expression of some particular markers. Nevertheless, populations of "no interest" such as the "NO_BTNK" are also present in the sample. Phenotypic differences without any significant biological interest can be detected by the clustering algorithms. This problem is aggravated with the increase in the number of parameters: markers that are used to define a specific population can reveal different expression patterns in other populations. Hence, increasing the number of parameters might increment the number of unexpected clusters. In this regard, the matching procedure developed in this study already merges the clusters matching to the same population. Thus, the procedure is particularly well adapted to benchmark algorithms used with high-dimensional experimental designs.

As a matter of fact, new methodologies must be accompanied by new conceptions. With the manual gating, populations are explored in a supervised manner. For example, in the 11-parameter experimental design, the CD27 marker is used to characterize B and T cell subpopulations. As it is not a useful marker to define the NKT cells, its expression in this cellular compartment is not explored. Nevertheless, NKT cells do express CD27 in a heterogenous manner. Researchers will thus need to improve their knowledge in the cell phenotype of the studied populations (at least for the markers used in the same experimental design) in order to correctly interpret the results obtained with unsupervised clustering algorithms. With respect to the benchmarking studies, special attention must be paid to the choice of reference populations.

Finally, there is the question about the limits of the methodology. New clustering algorithms for cytometry data are validated with the results obtained by manual gating, which is precisely the methodology that needs to get improved. Indeed, looking at the gating strategies performed on histograms or dot plots, it is noticeable that the boundaries between cell populations are often arbitrary. Thus, some of the discrepancies observed between the evaluated methods and the manual gated populations of reference might actually be due to this artefact. Further investigations based on synthetic datasets should help strengthen the evaluation process.

7 Conclusions

An extensive review on unsupervised clustering algorithms for high-dimensional data has been performed in order to select the algorithms that could potentially fulfill the requirements of the IRSL Flow Cytometry facility users. These algorithms have been benchmarked using a methodology developed to that end. This methodology includes:

- The production of synthetic data
- A methodology to label the cells from real cytometry experiences according to a manual gating strategy
- A performance evaluation method including a matching procedure specially designed for this project

This methodology has been tested with two different experimental designs for human leukocyte characterization using 5 and 11 parameters. The results obtained with these tests indicate that `RPhenograph` is the most adequate clustering method in terms of accuracy and computational time and thus the best candidate for the `Shiny` front-end for the IRSL Flow Cytometry facility. Finally, the methods developed during this work are ready to be conducted with other experimental designs including a higher number of parameters.

8 Glossary

IRSL Institut de Recherche Saint-Louis

B B lymphocyte

CPU Central Processor Unit

FCS Flow Cytometry Standard

FlowCAP Flow Cytometry: Critical Assessment of Population Identification Methods

NK Natural killer cell

NKT Natural killer T cell

NNG nearest neighbor graph

NO_BTNK No B, no T, no NK lymphocyte

SOM self-organizing map

T T lymphocyte

t-SNE t-Stochastic Neighbor Embedding

UMAP Uniform Manifold Approximation and Projection

9 References

1. Saeys Y, Van Gassen S, Lambrecht BN. Computational flow cytometry: Helping to make sense of high-dimensional immunology data. *Nature Reviews. Immunology.* 16(7), 449–462 (2016).

2. Mair F, Hartmann FJ, Mrdjen D, Tosevski V, Krieg C, Becher B. The end of gating? An introduction to automated analysis of high dimensional cytometry data. *European Journal of Immunology.* 46(1), 34–43 (2016).

3. Weber LM, 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.* 89(12), 1084–1096 (2016).

4. Platon L, Pejoski D, Gautreau G *et al.* A computational approach for phenotypic comparisons of cell populations in high-dimensional cytometry data. *Methods (San Diego, Calif.).* 132, 66–75 (2018).

5. Kimball AK, Oko LM, Bullock BL, Nemenoff RA, Dyk LF van, Clambey ET. A beginner's guide to analyzing and visualizing mass cytometry data. *The Journal of Immunology.* 200(1), 3–22 (2018).

6. Maaten L van der, Hinton G. Visualizing data using t-sne. *Journal of machine learning research.* 9(Nov), 2579–2605 (2008).

7. McInnes L, Healy J, Melville J. UMAP: Uniform manifold approximation and projection for dimension reduction. (2018). Available from: http://arxiv.org/abs/http://arxiv.org/abs/1802.03426v2.

8. Becht E, McInnes L, Healy J *et al.* Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology.* 37(1), 38–44 (2018).

9. Chen H, Lau MC, Wong MT, Newell EW, Poidinger M, Chen J. Cytofkit: A bioconductor package for an integrated mass cytometry data analysis pipeline. *PLOS Computational Biology.* 12(9), e1005112 (2016).

10. Aghaeepour N, Nikolic R, Hoos HH, Brinkman RR. Rapid cell population identification in flow cytometry data. *Cytometry Part A.* 79A(1), 6–13 (2010).

11. Ge Y, Sealfon SC. flowPeaks: A fast unsupervised clustering for flow cytometry data via k-means and density peak finding. *Bioinformatics.* 28(15), 2052–2058 (2012).

12. Qiu P, Simonds EF, Bendall SC *et al.* Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE. *Nature Biotechnology.* 29(10), 886–891 (2011).

13. Gassen SV, Callebaut B, Helden MJV *et al.* FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry Part A.* 87(7), 636–645 (2015).

14. Levine JH, Simonds EF, Bendall SC *et al.* Data-driven phenotypic dissection of AML

reveals progenitor-like cells that correlate with prognosis. *Cell.* 162(1), 184–197 (2015).

15. Aghaeepour N, Greg Finak, Hoos H *et al.* Critical assessment of automated flow cytometry data analysis techniques. *Nature Methods.* 10(3), 228–238 (2013).

16. Samusik N, Good Z, Spitzer MH, Davis KL, Nolan GP. Automated mapping of phenotype space with single-cell data. *Nature Methods.* 13(6), 493–496 (2016).

17. Ellis B, Haaland P, Hahne F *et al.* FlowCore: FlowCore: Basic structures for flow cytometry data..

18. Van P, Jiang W, Gottardo R, Finak G. ggCyto: Next generation open-source visualization software for cytometry. *Bioinformatics.* 34(22), 3951–3953 (2018).

19. Finak G, Jiang M. FlowWorkspace: Infrastructure for representing and interacting with gated and ungated cytometry data sets..

20. Finak G, Jiang W, Gottardo R. CytoML for cross-platform cytometry data sharing. *Cytometry Part A.* 93(12), 1189–1196 (2018).

21. Gianni Monaco CH. FlowAI. (2017).

22. Fletez-Brant K, Spidlen J, Brinkman RR, Roederer M, Chattopadhyay PK. flowClean: Automated identification and removal of fluorescence anomalies in flow cytometry data. *Cytometry Part A.* (2016).

23. Amir E-aD, Davis KL, Tadmor MD *et al.* viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature Biotechnology.* 31(6), 545–552 (2013).

24. Unen V van, Höllt T, Pezzotti N *et al.* Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types. *Nature Communications.* 8(1) (2017).

25. Shekhar K, Brodin P, Davis MM, Chakraborty AK. Automatic classification of cellular expression by nonlinear stochastic embedding (ACCENSE). *Proceedings of the National Academy of Sciences.* 111(1), 202–207 (2013).

26. Becher B, Schlitzer A, Chen J *et al.* High-dimensional analysis of the murine myeloid cell system. *Nature Immunology.* 15(12), 1181–1189 (2014).

27. Lo K, Hahne F, Brinkman RR, Gottardo R. flowClust: A bioconductor package for automated gating of flow cytometry data. *BMC Bioinformatics.* 10(1) (2009).

28. Finak G, Bashashati A, Brinkman R, Gottardo R. Merging mixture components for cell population identification in flow cytometry. *Advances in Bioinformatics.* 2009, 1–12 (2009).

29. Pyne S, Hu X, Wang K *et al.* Automated high-dimensional flow cytometric data analysis. *Proceedings of the National Academy of Sciences.* 106(21), 8519–8524 (2009).

30. Qian Y, Wei C, Lee FE-H *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.* 78B(S1), S69–S82 (2010).

31. Zare H, Shooshtari P, Gupta A, Brinkman RR. Data reduction for spectral clustering to analyze high throughput flow cytometry data. *BMC Bioinformatics.* 11(1), 403 (2010).

32. Naim I, Datta S, Rebhahn J, Cavenaugh JS, Mosmann TR, Sharma G. SWIFT-scalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, part 1: Algorithm design. *Cytometry Part A.* 85(5), 408–421 (2014).

33. Sörensen T, Baumgart S, Durek P, Grützkau A, Häupl T. immunoClust-an automated analysis pipeline for the identification of immunophenotypic signatures in high-dimensional cytometric datasets. *Cytometry Part A.* 87(7), 603–615 (2015).

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

35. ONeill K, Jalali A, Aghaeepour N, Hoos H, Brinkman RR. Enhanced flowType/RchyOptimyx: A bioconductor pipeline for discovery in high-dimensional cytometry data. *Bioinformatics.* 30(9), 1329–1330 (2014).

36. Malek M, Taghiyar MJ, Chong L, Finak G, Gottardo R, Brinkman RR. flowDensity: Reproducing manual gating of flow cytometry data by automated density-based cell population identification. *Bioinformatics.* 31(4), 606–607 (2014).

37. Aghaeepour N, Simonds EF, Knapp DJHF *et al.* GateFinder: Projection-based gating strategy optimization for flow and mass cytometry. *Bioinformatics.* 34(23), 4131–4133 (2018).

38. Bruggner RV, Bodenmiller B, Dill DL, Tibshirani RJ, Nolan GP. Automated identification of stratifying signatures in cellular subpopulations. *Proceedings of the National Academy of Sciences.* 111(26), E2770–E2777 (2014).

39. Cheng Y, Wong MT, Maaten L van der, Newell EW. Categorical analysis of human t cell heterogeneity with one-dimensional soli-expression by nonlinear stochastic embedding. *The Journal of Immunology.* 196(2), 924–932 (2015).

40. Zunder ER, Lujan E, Goltsev Y, Wernig M, Nolan GP. A continuous molecular roadmap to iPSC reprogramming through progression analysis of single-cell mass cytometry. *Cell stem cell.* 16(3), 323–337 (2015).

41. Johnsson K, Wallin J, Fontes M. BayesFlow: Latent modeling of flow cytometry cell populations. *BMC Bioinformatics.* 17(1) (2016).

42. Beyrend G, Stam K, Höllt T, Ossendorp F, Arens R. Cytofast: A workflow for visual and quantitative analysis of flow and mass cytometry data to discover immune signatures and correlations. *Computational and Structural Biotechnology Journal.* 16, 435–442 (2018).

43. Shen Y, Chaigne-Delalande B, Lee RWJ, Losert W. CytoBinning: Immunological insights from multi-dimensional data. *PLOS ONE.* 13(10), e0205291 (2018).